Java学习群

72030155

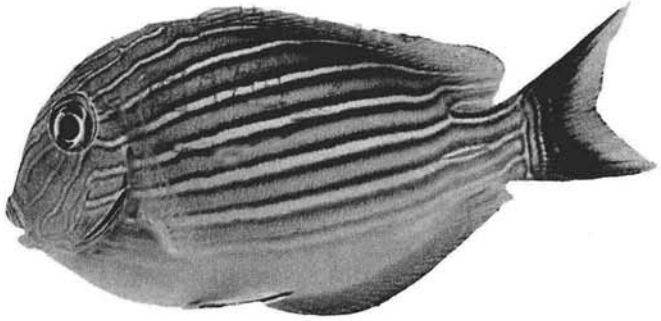
进群可以免费获取视频教程以及每日免费听老师讲课

Java学习群：72030155

好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Java Web
入门经典

王国辉 陈英 ◎等编著



机械工业出版社
China Machine Press

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

图书在版编目（CIP）数据

Java Web 入门经典 / 王国辉等著 . —北京：机械工业出版社，2013.4
（华章程序员书库）

ISBN 978-7-111-42182-5

Ⅰ. J… Ⅱ. 王… Ⅲ. JAVA 语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2013）第 077264 号

版权所有 · 侵权必究
封底无防伪标均为盗版
本书法律顾问 北京市展达律师事务所

本书以初学者为核心，全面介绍了 Java Web 开发中常用的各种技术。内容排列上由浅入深，让读者循序渐进掌握编程技术；在内容讲解上结合丰富的图解和形象的比喻，帮助读者理解“晦涩难懂”的技术；在内容形式上附有大量的提示、技巧、说明等栏目，夯实读者编程技术，丰富编程经验。全书共分 4 篇 19 章，其中，第一篇为“起步篇”，主要包括开启 Java Web 之门、不可不知的客户端应用技术、驾驭 Java Web 开发环境、Java Web 开发必修课之 JSP 语法等内容；第二篇为“核心篇”，主要包括程序开发效率之利器——内置对象、完美的“咖啡豆”——JavaBean 技术、Web 应用的缔造者——Servlet、使用表达式语言（EL）、页面控制利器——JSTL 标签库、数据库应用开发等内容；第三篇为“高级应用篇”，主要包括在互联网中操作文件资源、利用 JFreeChart 实现动态图表、在 JSP 中操作 XML、完美体验 Ajax 技术、MVC 架构实现者——Struts 2 框架、数据库持久化利器——Hibernate 技术、简化企业开发瑰宝——Spring 技术等内容；第四篇为“项目实战篇”，主要包括基于 Ajax 的聊天室和基于 SSH2 的网络商城两个实战项目。

本书附有配套光盘，光盘中提供有本书所有程序的源代码、多媒体教学视频、程序开发资源库。其中，源代码全部经过精心测试，能够在 Windows XP、Windows Server 2003、Windows 7 操作系统上编译和运行。

本书适用于 Java Web 的爱好者、初学者和中级开发人员，也可以作为大中专院校和培训机构的教材。

机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码 100037）
责任编辑：高婧雅
北京京师印务有限公司印刷
2013 年 6 月第 1 版第 1 次印刷
185mm × 260mm • 32.75 印张
标准书号：ISBN 978-7-111-42182-5
ISBN 978-7-89433-903-4（光盘）
定 价：69.00 元（附光盘）

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

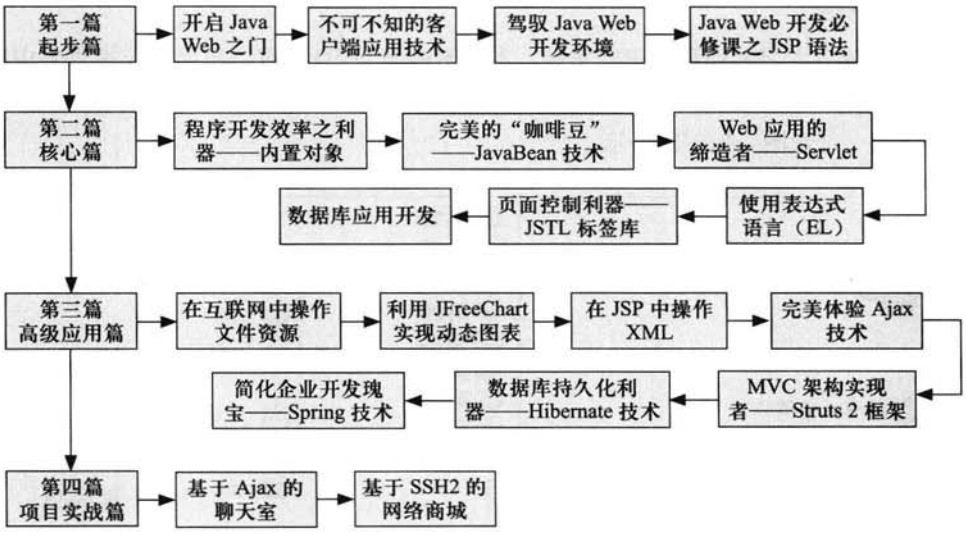
前　言

Java 是当前程序开发中最流行的编程语言之一，其从诞生以来就受到了广大程序开发人员的追捧，它可以开发桌面应用、网站程序、手机和电子设备应用程序，等等，特别在 Web 程序开发方面，Java 的跨平台优越性体现得更加淋漓尽致。近年来，Java Web 框架技术层出不穷，更体现了 Java 在 Web 方面强大的生命力。

目前，有关 Java Web 的书籍有很多，但是真正从初学者的角度出发，把技术及应用讲解透彻的并不是很多。本书从初学者的角度出发，为想要学习 Java Web 编程、想要进行 Java Web 开发的初中级开发人员、编程爱好者、大学师生精心策划。所讲内容从技术应用的角度出发，结合实际应用进行讲解，以便让想要学习 Java Web 编程的开发人员能够学会编程，并能够根据自己的意愿开发出实用的软件。

本书内容

本书按照学习 Java Web 开发的最佳结构来分配章节，共分为 4 篇 19 章内容，其中，前三篇可以使读者掌握 Java Web 的编程知识，第四篇可以使读者学习实际项目的开发过程。本书知识结构如下图所示。



本书特点

❑ 循序渐进，简单易学

本书以初中级程序员为对象，先从了解 Java Web 和搭建开发环境学起，再学习 Java Web

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

开发的基础技术，然后学习 Java Web 开发的高级内容，最后学习两个完整项目的开发。讲解过程中步骤详尽，版式新颖，让读者在阅读时一目了然，从而快速掌握书中内容。

❑ 学练结合，注释详尽

本书在讲解知识时，提供了丰富的实例，帮助读者进行随学随练，真正做到学有所用。另外，为了便于读者更好地学习和使用本书，书中所有的代码都提供了详尽的注释。

❑ 上机实践，巩固知识

要充分理解和掌握一项技术，最佳的方式就是多练习、多实践。在介绍完每章所需内容后，本书还提供了多个上机实践供读者自己完成，以此来检验读者对本章内容的掌握情况。每个实践任务分为“题目描述”和“技术指导”两部分。其中，“题目描述”给出了实例功能的详细描述和效果图，“技术指导”则给出了实现实例的思路和关键代码。

❑ 视频学习，讲解详尽

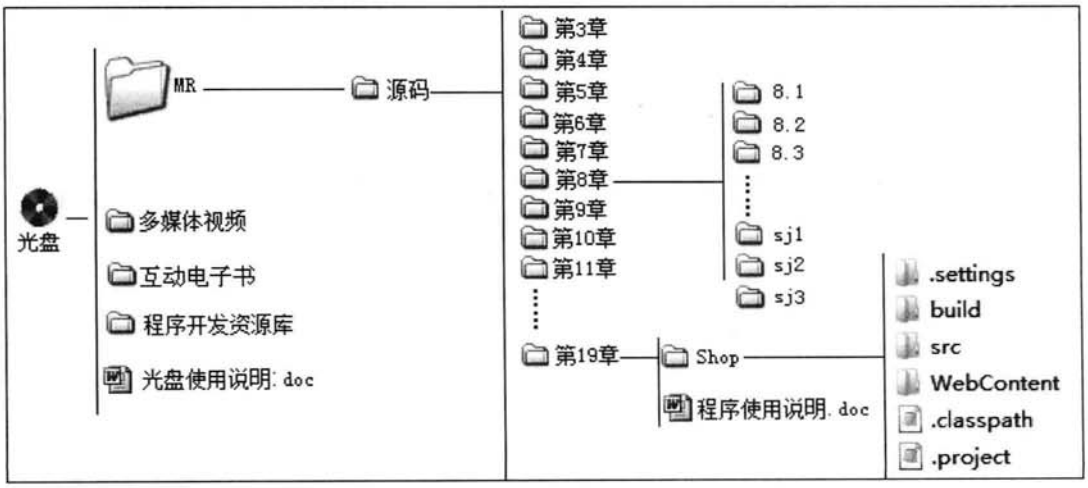
为了更方便读者学习书中的内容，本书在配套光盘中提供了 14 小时的多媒体视频讲座，其讲解细腻、层次清楚、互动性强。在加深对内容的理解同时，切实解决“是什么”、“为什么”、“怎么办”、“应注意些什么”等内容。

❑ 海量资源，超值奉送

本书光盘中提供了明日科技数十年积累的程序开发资源库，包括海量实例资源库、模块资源库、项目案例资源库、测试题库系统、面试资源库等内容。

超值 DVD 光盘

为了帮助读者学习和使用书中的实例，本书附赠有 DVD 光盘，里面不仅提供书中所有实例项目的源代码、素材、光盘使用说明书，还提供了 14 小时多媒体教学视频及程序开发资源库。光盘目录（示意图）如下图所示。



Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

读者对象

本书是一部零基础 Java Web 编程图书，非常适合以下人员阅读：

- 零基础的初学编程人员
- 初中级程序开发人员
- 编程爱好者
- 想学习编程的各企事业单位在职人员
- 大中专院校的老师和学生
- 相关培训机构的老师和学员

读者服务

秉着“十年服务，始终如一”的理念，明日科技承诺，如果你在学习或使用本书的过程中遇到问题或疑惑，可以通过如下方式与我们联系：

- 服务网站：www.mingribook.com
- 服务 QQ：4006751066
- 服务信箱：mingrisoft@mingrisoft.com
- 服务电话：0431-84978981/84978982
- 学习社区：www.mrbccd.com

我们将在 5 个工作日内为你提供满意的回答。

本书作者

本书由明日科技组织编写，参加编写的有王国辉、陈英、陈丹丹、王小科、王雨竹、隋光宇、杨丽、顾彦玲、宋禹蒙、高春艳、刘莉莉、赛奎春等。由于作者水平有限，疏漏和不足之处在所难免，欢迎广大读者提出宝贵意见。

非学无以广才，非志无以成学。祝大家读书快乐！

明日科技

2012 年 8 月

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

目　录

前言

第一篇　起步篇

第 1 章　开启 Jave Web 之门 …… 2
视频讲解：19 分钟
1.1　初识 Java Web …… 2
1.1.1　Java 概述 …… 2
1.1.2　Java 语言的主要特点 …… 3
1.1.3　Java Web 概述 …… 3
1.1.4　Java Web 成功案例 …… 4
1.2　Web 应用技术 …… 5
1.2.1　客户端应用技术 …… 5
1.2.2　服务器端应用技术 …… 7
1.3　Web 应用程序的工作原理 …… 8
1.4　网络程序开发体系结构 …… 8
1.4.1　什么是 C/S 结构 …… 9
1.4.2　什么是 B/S 结构 …… 9
1.4.3　B/S 与 C/S 的区别 …… 9
1.5　Java Web 网站的基本构建流程 …… 10
1.6　常用网上资源 …… 10
1.6.1　常用资源下载网 …… 10
1.6.2　技术社区 …… 11
1.7　小结 …… 11
1.8　习题 …… 11
第 2 章　不可不知的客户端应用技术 …… 12
视频讲解：113 分钟
2.1　构建页面内容的 HTML 5 标记语言 …… 12
2.1.1　了解 HTML 5 文档结构 …… 12

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2.1.2 HTML 文字排版标记 …… 14
2.1.3 图片与超链接标记 …… 16
2.1.4 HTML 5 新增的语义元素 …… 18
2.1.5 制作表格 …… 21
2.1.6 播放音频和视频 …… 22
2.1.7 表单标记 …… 24
2.2 美化页面的 CSS 样式表 …… 28
2.2.1 样式表的定义与引用 …… 28
2.2.2 CSS 规则 …… 30
2.2.3 CSS 选择器 …… 30
2.2.4 CSS 常用属性 …… 32
2.3 客户端交互的 JavaScript …… 34
2.3.1 JavaScript 的语言基础 …… 34
2.3.2 JavaScript 的流程控制语句 …… 38
2.3.3 JavaScript 函数的定义及调用 …… 41
2.3.4 事件处理 …… 41
2.3.5 常用对象 …… 43
2.3.6 DOM 技术 …… 46
2.4 上机实践 …… 49
2.4.1 使用删除线样式标注商品特价 …… 49
2.4.2 将小写金额转换为大写金额 …… 49
2.4.3 验证用户注册信息的合法性 …… 50
2.5 小结 …… 51
2.6 习题 …… 51
第 3 章 驾驭 Java Web 开发环境 …… 53
视频讲解：25 分钟
3.1 Java Web 所需要的开发环境 …… 53
3.1.1 开发工具包 JDK …… 53
3.1.2 Web 服务器 …… 54
3.1.3 数据库 …… 54
3.1.4 Web 浏览器 …… 55
3.2 安装和配置 JDK …… 55
3.2.1 下载 JDK …… 55
3.2.2 安装 JDK …… 56
3.2.3 在 Windows 系统下配置和测试 JDK …… 57
3.3 安装和配置 Tomcat …… 58
3.3.1 获取 Tomcat …… 59
3.3.2 熟悉 Tomcat …… 60

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

3.3.3 修改 Tomcat 的默认端口 61
3.4 Eclipse 开发工具的安装与使用 62
3.4.1 Eclipse 的下载与安装 62
3.4.2 启动 Eclipse 62
3.4.3 安装 Eclipse 中文语言包 63
3.4.4 Eclipse 工作台 63
3.5 做好项目开发的准备工作 64
3.5.1 集成 Eclipse 与 Tomcat 64
3.5.2 完善 Web 项目开发所需配置 65
3.6 使用 Eclipse 开发一个 JSP 网站 66
3.7 小结 68
3.8 习题 69
第 4 章 Java Web 开发必修课之 JSP 语法 70
视频讲解：55 分钟
4.1 了解 JSP 页面的基本构成 70
4.2 使用 JSP 指令标识 71
4.2.1 使用页面指令 page 72
4.2.2 使用文件包含指令 include 74
4.2.3 使用引用标签库指令 taglib 75
4.3 使用 JSP 脚本 76
4.3.1 在 JSP 中应用代码片段 76
4.3.2 使用 JSP 表达式 77
4.3.3 使用声明标识 77
4.4 巧用注释 78
4.4.1 基本 HTML 注释 78
4.4.2 隐藏注释 79
4.4.3 代码片段中的注释 79
4.5 常用 JSP 动作标识 80
4.5.1 使用包含动作标识 <jsp:include> 81
4.5.2 使用请求转发的动作标识 <jsp:forward> 83
4.5.3 使用子动作标识 <jsp:param> 84
4.5.4 使用动作标识 <jsp:useBean> 85
4.5.5 使用动作标识 <jsp:getProperty> 91
4.5.6 使用动作标识 <jsp:setProperty> 91
4.6 上机实践 93
4.6.1 应用 Java 程序片段动态生成表格 93
4.6.2 将页面转发到用户登录页面 94
4.6.3 应用 JavaBean 保存图片信息 95

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

4.7 小结 …… 96
4.8 习题 …… 96

第二篇 核心篇

第 5 章 程序开发效率之利器——内置对象 …… 98
视频讲解：69 分钟
5.1 JSP 内置对象概述 …… 98
5.1.1 内置对象的作用 …… 98
5.1.2 内置对象及其应用场合 …… 98
5.2 request 请求对象 …… 99
5.2.1 获取请求参数 …… 99
5.2.2 解决中文乱码问题 …… 101
5.2.3 获取客户端数据 …… 101
5.2.4 应用 request 对象域 …… 103
5.3 response 响应对象 …… 104
5.3.1 操作 HTTP 头信息 …… 104
5.3.2 设置 MIME 类型 …… 105
5.3.3 实现页面重定向 …… 105
5.4 out 输出对象 …… 106
5.4.1 向客户端输出数据 …… 106
5.4.2 管理缓冲区 …… 107
5.5 session 会话对象 …… 107
5.5.1 session 的生命周期 …… 108
5.5.2 创建与获取会话 …… 108
5.5.3 移除对话中的数据 …… 108
5.5.4 设置会话的有效时间 …… 109
5.5.5 应用 session 对象模拟用户登录 …… 109
5.6 应用 application 对象 …… 111
5.6.1 application 对象的生命周期 …… 111
5.6.2 操作 application 对象中数据 …… 111
5.6.3 配置 Web 应用的初始化参数 …… 113
5.6.4 应用 application 对象实现网页计数器 …… 114
5.7 其他内置对象 …… 114
5.7.1 应答与请求的 page 对象 …… 114
5.7.2 页面上下文的 pageContext 对象 …… 115
5.7.3 获取 web.xml 配置信息的 config 对象 …… 115
5.7.4 获取异常信息的 exception 对象 …… 116
5.8 上机实践 …… 116

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

5.8.1 通过 request 对象获取用户注册信息 116
5.8.2 防止表单在网站外部提交 117
5.8.3 通过 cookie 保存并读取用户登录信息 118
5.9 小结 119
5.10 习题 119
第 6 章 完美的“咖啡豆”——JavaBean 技术 120
视频讲解：26 分钟
6.1 JavaBean 概述 120
6.1.1 JavaBean 的产生背景 120
6.1.2 JavaBean 的作用 120
6.1.3 JavaBean 的规范 121
6.1.4 JavaBean 的种类 121
6.2 Eclipse 对 JavaBean 方法的支持 122
6.3 JavaBean 的应用 124
6.3.1 获取 JavaBean 的属性 124
6.3.2 对 JavaBean 的属性赋值 126
6.3.3 如何在 JSP 页面中应用 JavaBean 127
6.4 上机实践 130
6.4.1 判断用户名是否有效 130
6.4.2 计算两个日期相差的天数 131
6.4.3 随机生成指定位数的验证码 131
6.5 小结 132
6.6 习题 132
第 7 章 Web 应用的缔造者——Servlet 134
视频讲解：46 分钟
7.1 Servlet 技术 134
7.1.1 Servlet 概述 134
7.1.2 Servlet 功能 134
7.1.3 Servlet 特点 135
7.1.4 Servlet 的生命周期 136
7.2 Servlet 技术开发 136
7.2.1 Servlet 在 Java EE 中的结构体系 136
7.2.2 Servlet 核心 API 137
7.2.3 创建第一个 Servlet 类 138
7.2.4 Servlet 配置 139
7.2.5 使用 Servlet 处理业务逻辑 140
7.2.6 使用 Eclipse 快速创建 Servlet 141
7.3 Servlet 的典型应用 144
7.3.1 处理表单数据 144

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

7.3.2 页面转发 …… 147
7.3.3 获取当前页的绝对路径 …… 148
7.4 配置并使用 Servlet 过滤器 …… 149
7.4.1 过滤器的处理方式 …… 149
7.4.2 过滤器的核心对象 …… 149
7.4.3 创建并配置过滤器 …… 150
7.4.4 字符编码过滤器 …… 153
7.5 Servlet 监听器 …… 157
7.5.1 Servlet 监听器简介 …… 157
7.5.2 Servlet 监听器的工作原理 …… 158
7.5.3 监听 Servlet 上下文 …… 158
7.5.4 监听 HTTP 会话 …… 158
7.5.5 监听 Servlet 请求 …… 159
7.5.6 使用监听器查看在线用户 …… 160
7.6 上机实践 …… 162
7.6.1 记录用户访问次数 …… 162
7.6.2 防盗链过滤器 …… 162
7.6.3 敏感词过滤器 …… 163
7.7 小结 …… 164
7.8 习题 …… 164
第 8 章 使用表达式语言（EL） …… 165
视频讲解：72 分钟
8.1 EL 概述 …… 165
8.1.1 EL 的基本语法 …… 165
8.1.2 EL 的特点 …… 166
8.2 与低版本的环境兼容——禁用 EL …… 166
8.2.1 使用斜杠符号“\” …… 166
8.2.2 使用 page 指令 …… 166
8.2.3 在 web.xml 文件中配置 <el-ignored> 元素 …… 167
8.3 保留关键字 …… 167
8.4 EL 运算符及其优先级 …… 168
8.4.1 通过 EL 访问数据 …… 168
8.4.2 在 EL 中执行算术运算 …… 170
8.4.3 在 EL 中判断对象是否为空 …… 170
8.4.4 在 EL 中执行逻辑关系运算 …… 171
8.4.5 在 EL 中执行条件运算 …… 172
8.5 EL 的隐含对象 …… 173
8.5.1 页面上下文对象 …… 173
8.5.2 访问作用域范围的隐含对象 …… 175

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

8.5.3 访问环境信息的隐含对象 …… 176
8.6 定义和使用 EL 函数 …… 179
8.6.1 定义和使用函数 …… 179
8.6.2 定义和使用 EL 函数时常见的错误 …… 181
8.7 上机实践 …… 182
8.7.1 应用 EL 访问 JavaBean 属性 …… 182
8.7.2 应用 EL 表达式显示投票结果 …… 183
8.7.3 应用 EL 显示用户的登录状态 …… 184
8.8 小结 …… 184
8.9 习题 …… 185
第 9 章 页面控制利器——JSTL 标签库 …… 186
视频讲解：85 分钟
9.1 JSTL 标签库简介 …… 186
9.2 JSTL 的下载与配置 …… 187
9.2.1 下载 JSTL 标签库 …… 188
9.2.2 配置 JSTL …… 188
9.3 表达式标签 …… 190
9.3.1 <c:out> 输出标签 …… 190
9.3.2 <c:set> 变量设置标签 …… 192
9.3.3 <c:remove> 变量移除标签 …… 194
9.3.4 <c:catch> 捕获异常标签 …… 195
9.4 URL 相关标签 …… 197
9.4.1 <c:import> 导入标签 …… 197
9.4.2 <c:url> 动态生成 URL 标签 …… 199
9.4.3 <c:redirect> 重定向标签 …… 200
9.4.4 <c:param> 传递参数标签 …… 201
9.5 流程控制标签 …… 202
9.5.1 <c:if> 条件判断标签 …… 202
9.5.2 <c:choose> 条件选择标签 …… 203
9.5.3 <c:when> 条件测试标签 …… 204
9.5.4 <c:otherwise> 其他条件标签 …… 205
9.6 循环标签 …… 206
9.6.1 <c:forEach> 循环标签 …… 207
9.6.2 <c:forTokens> 迭代标签 …… 209
9.7 上机实践 …… 210
9.7.1 利用 JSTL 标签导入用户注册协议 …… 210
9.7.2 根据参数请求显示不同的页面 …… 211
9.7.3 JSTL 在电子商城中的应用 …… 211
9.8 小结 …… 213

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

9.9 习题 …… 213
第 10 章 数据库应用开发 …… 214
视频讲解：61 分钟
10.1 Web 开发时常用的数据库 …… 214
10.1.1 MySQL 数据库 …… 214
10.1.2 SQL Server 2008 …… 219
10.1.3 Oracle 数据库 …… 219
10.2 JDBC 技术 …… 219
10.2.1 JDBC 简介 …… 219
10.2.2 JDBC 连接数据库的开发流程 …… 220
10.3 JDBC 核心 API …… 221
10.3.1 驱动程序管理器 DriverManager …… 221
10.3.2 代表数据库连接的 Connection 接口 …… 223
10.3.3 执行 SQL 语句的 Statement 接口 …… 223
10.3.4 预编译执行 SQL 语句的 PreparedStatement 接口 …… 225
10.3.5 结果集 ResultSet 接口 …… 226
10.3.6 及时关闭连接 …… 227
10.4 通过 JDBC 操作数据库 …… 228
10.4.1 添加数据 …… 228
10.4.2 查询数据 …… 230
10.4.3 修改数据 …… 233
10.4.4 删除数据 …… 235
10.4.5 批处理 …… 237
10.4.6 调用存储过程 …… 240
10.5 JDBC 在 Java Web 中的应用 …… 242
10.5.1 开发模式 …… 242
10.5.2 分页查询 …… 243
10.6 上机实践 …… 248
10.6.1 批量删除数据 …… 248
10.6.2 网站用户注册 …… 249
10.6.3 插入用户登录日志信息 …… 250
10.7 小结 …… 251
10.8 习题 …… 251

第三篇 高级篇

第 11 章 在互联网中操作文件资源 …… 254
视频讲解：83 分钟
11.1 使用 Servlet 3.0 的新特性实现文件上传 …… 254

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

11.2 使用 Commons-FileUpload 组件实现文件上传 255
11.2.1 添加表单及表单元素 256
11.2.2 创建上传对象 256
11.2.3 解析上传请求 256
11.3 实现文件下载 259
11.3.1 文件类 259
11.3.2 文件字节输入流 260
11.3.3 字节输出流 261
11.4 上机实践 262
11.4.1 获取表单的元素值 262
11.4.2 限制文件上传的类型 263
11.4.3 批量文件上传 264
11.5 小结 265
11.6 习题 265
第 12 章 利用 JFreeChart 实现动态图表 267
视频讲解：39 分钟
12.1 JFreeChart 简介 267
12.1.1 认识 JFreeChart 组件 267
12.1.2 JFreeChart 的下载与使用 268
12.2 JFreeChart 的核心对象 269
12.2.1 制图对象 269
12.2.2 制图工厂对象 269
12.2.3 数据集合对象 270
12.2.4 绘图区对象 271
12.2.5 坐标轴对象 273
12.2.6 图片渲染对象 274
12.3 JFreeChart 的应用 275
12.3.1 如何获取图片 275
12.3.2 创建数据集合与 JFreeChart 实例 276
12.3.3 图表相关属性的设置 278
12.3.4 JFreeChart 内置 JDBC 的使用 280
12.3.5 中文乱码的解决方案 283
12.4 上机实践 284
12.4.1 利用 JFreeChart 绘制折线图 284
12.4.2 利用 JFreeChart 绘制区域图 285
12.4.3 利用 JFreeChart 绘制时序图 286
12.5 小结 288
12.6 习题 288

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 13 章 在 JSP 中操作 XML ····· 289
视频讲解：29 分钟
13.1 XML 简介 ····· 289
13.1.1 XML 文档结构 ····· 289
13.1.2 XML 语法要求 ····· 291
13.1.3 为 XML 文档中的元素定义属性 ····· 291
13.1.4 XML 的注释 ····· 291
13.1.5 处理字符数据 ····· 291
13.2 dom4j 概述 ····· 293
13.2.1 dom4j 简介 ····· 293
13.2.2 dom4j 的下载与配置 ····· 293
13.3 创建 XML 文件 ····· 295
13.3.1 创建 XML 文档对象 ····· 295
13.3.2 创建根节点 ····· 296
13.3.3 添加注释 ····· 296
13.3.4 添加属性 ····· 297
13.3.5 创建子节点 ····· 297
13.3.6 设置节点的内容 ····· 297
13.3.7 设置编码 ····· 298
13.3.8 设置输出格式 ····· 298
13.3.9 输出 XML 文件 ····· 298
13.4 解析 XML 文档 ····· 300
13.4.1 构建 XML 文档对象 ····· 300
13.4.2 获取根节点 ····· 301
13.4.3 获取子节点 ····· 301
13.5 修改 XML 文档 ····· 302
13.5.1 修改节点 ····· 302
13.5.2 删除节点 ····· 303
13.6 上机实践 ····· 304
13.6.1 保存公告信息到 XML 文件 ····· 304
13.6.2 遍历 XML 文档 ····· 305
13.6.3 管理保存在 XML 文件中的公告信息 ····· 307
13.7 小结 ····· 307
13.8 习题 ····· 308
第 14 章 完美体验 Ajax 技术 ····· 309
视频讲解：34 分钟
14.1 了解 Ajax ····· 309
14.1.1 什么是 Ajax ····· 309

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

14.1.2 Ajax 开发模式与传统开发模式的比较 …… 309
14.1.3 Ajax 的优点 …… 310
14.2 使用 XMLHttpRequest 对象 …… 310
14.2.1 初始化 XMLHttpRequest 对象 …… 310
14.2.2 XMLHttpRequest 对象的常用方法 …… 311
14.2.3 XMLHttpRequest 对象的常用属性 …… 313
14.3 传统 Ajax 的工作流程 …… 314
14.3.1 发送请求 …… 314
14.3.2 处理服务器响应 …… 316
14.3.3 一个完整的实例——检测用户名是否唯一 …… 317
14.4 利用 jQuery 实现 Ajax …… 319
14.4.1 jQuery 简介 …… 320
14.4.2 应用 load() 方法发送请求 …… 322
14.4.3 发送 GET 和 POST 请求 …… 324
14.4.4 服务器返回的数据格式 …… 329
14.4.5 使用 $.ajax() 方法 …… 332
14.5 需要注意的几个问题 …… 334
14.5.1 安全问题 …… 334
14.5.2 性能问题 …… 335
14.5.3 浏览器兼容性问题 …… 335
14.5.4 中文编码问题 …… 336
14.6 上机实践 …… 336
14.6.1 实时显示公告信息 …… 336
14.6.2 实现无刷新分页 …… 337
14.6.3 多级联动下拉列表 …… 338
14.7 小结 …… 338
14.8 习题 …… 339
第 15 章 MVC 架构实现者——Struts 2 框架 …… 340
视频讲解：93 分钟
15.1 Struts 2 框架概述 …… 340
15.1.1 理解 MVC 的原理 …… 340
15.1.2 Struts 2 框架的产生 …… 341
15.1.3 Struts 2 的结构体系 …… 341
15.2 Struts 2 入门 …… 342
15.2.1 获取与配置 Struts 2 …… 342
15.2.2 创建第 1 个 Struts 2 程序 …… 343
15.3 Action 对象 …… 345
15.3.1 认识 Action 对象 …… 345

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

15.3.2 请求参数的注入原理 346
15.3.3 Action 的基本流程 346
15.3.4 动态 Action 347
15.3.5 应用动态 Action 347
15.4 Struts 2 的配置文件 349
15.4.1 Struts 2 的配置文件类型 349
15.4.2 配置 Struts 2 包 350
15.4.3 配置名称空间 350
15.4.4 Action 的相关配置 351
15.4.5 使用通配符简化配置 352
15.4.6 配置返回结果 353
15.5 Struts 2 的标签库 354
15.5.1 应用数据标签 354
15.5.2 应用控制标签 356
15.5.3 应用表单标签 358
15.6 Struts 2 的开发模式 359
15.6.1 实现与 Servlet API 的交互 359
15.6.2 域模型 DomainModel 360
15.6.3 驱动模型 ModelDriven 361
15.7 Struts 2 的拦截器 363
15.7.1 拦截器概述 363
15.7.2 拦截器 API 363
15.7.3 使用拦截器 364
15.8 数据验证机制 366
15.8.1 手动验证 366
15.8.2 验证文件的命名规则 366
15.8.3 验证文件的编写风格 367
15.9 上机实践 368
15.9.1 应用 Struts 2 处理表单数据 368
15.9.2 Struts 2 标签下的用户注册 368
15.9.3 通过 Struts 2 实现简单投票器 369
15.10 小结 369
15.11 习题 369
第 16 章 数据库持久化利器——Hibernate 技术 371
视频讲解：85 分钟
16.1 初识 Hibernate 371
16.1.1 ORM 原理 371
16.1.2 Hibernate 简介 372

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

16.2 Hibernate 入门 ····· 372
16.2.1 获取 Hibernate ····· 372
16.2.2 Hibernate 配置文件 ····· 373
16.2.3 编写持久化类 ····· 374
16.2.4 Hibernate 映射 ····· 375
16.2.5 Hibernate 主键策略 ····· 376
16.3 使用 Hibernate 进行数据持久化 ····· 376
16.3.1 Hibernate 实例状态 ····· 377
16.3.2 Hibernate 初始化类 ····· 377
16.3.3 保存数据 ····· 379
16.3.4 查询数据 ····· 381
16.3.5 删除数据 ····· 382
16.3.6 修改数据 ····· 383
16.3.7 延迟加载 ····· 383
16.4 使用 Hibernate 的缓存 ····· 384
16.4.1 使用一级缓存 ····· 384
16.4.2 配置和使用二级缓存 ····· 385
16.5 实体关联关系映射 ····· 387
16.5.1 数据模型与领域模型 ····· 387
16.5.2 理解并配置多对一单向关联 ····· 388
16.5.3 理解并配置多对一双向关联 ····· 389
16.5.4 理解并配置一对一主键关联 ····· 391
16.5.5 级联操作 ····· 393
16.6 Hibernate 查询语言 ····· 394
16.6.1 HQL 语言 ····· 394
16.6.2 查询实体对象 ····· 395
16.6.3 条件查询 ····· 396
16.6.4 HQL 参数绑定机制 ····· 396
16.6.5 排序查询 ····· 397
16.6.6 应用聚合函数 ····· 397
16.6.7 分组方法 ····· 397
16.6.8 联合查询 ····· 398
16.7 上机实践 ····· 399
16.7.1 修改员工信息 ····· 399
16.7.2 员工信息的模糊查询 ····· 400
16.7.3 按年龄升序显示公民信息 ····· 400
16.8 小结 ····· 401
16.9 习题 ····· 401

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 17 章 简化企业开发瑰宝——Spring 技术 ····· 402
视频讲解：72 分钟
17.1 Spring 概述 ····· 402
17.1.1 Spring 的组成 ····· 402
17.1.2 获取 Spring ····· 403
17.1.3 配置 Spring ····· 403
17.1.4 使用 BeanFactory 管理 Bean ····· 405
17.1.5 应用 AplicationContext ····· 406
17.2 依赖注入 ····· 407
17.2.1 控制反转与依赖注入 ····· 407
17.2.2 配置 Bean ····· 408
17.2.3 Setter 注入 ····· 408
17.2.4 构造器注入 ····· 409
17.2.5 引用其他 Bean ····· 411
17.2.6 创建匿名内部 JavaBean ····· 412
17.3 AOP 概述 ····· 412
17.3.1 AOP 术语 ····· 413
17.3.2 AOP 的简单实现 ····· 414
17.4 Spring 的切入点 ····· 415
17.4.1 静态与动态切入点 ····· 415
17.4.2 深入静态切入点 ····· 416
17.4.3 深入切入点底层 ····· 417
17.4.4 Spring 中的其他切入点 ····· 417
17.5 Aspect 对 AOP 的支持 ····· 418
17.5.1 Aspect 概述 ····· 418
17.5.2 Spring 中的 Aspect ····· 418
17.5.3 DefaultPointcutAdvisor 切入点配置器 ····· 419
17.5.4 NameMatchMethodPointcutAdvisor 切入点配置器 ····· 420
17.6 Spring 的持久化 ····· 420
17.6.1 DAO 模式 ····· 420
17.6.2 Spring 的 DAO 理念 ····· 421
17.6.3 事务管理 ····· 423
17.6.4 应用 JdbcTemplate 操作数据库 ····· 426
17.6.5 与 Hibernate 整合 ····· 428
17.6.6 整合 Spring 与 Hibernate ····· 429
17.7 上机实践 ····· 430
17.7.1 登录页面国际化 ····· 430
17.7.2 利用 Spring 整合 Hibernate 来批量添加数据 ····· 431

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

17.7.3 利用 AOP 实现用户注册 ····· 431
17.8 小结 ····· 431
17.9 习题 ····· 432

第四篇 项目实战篇

第 18 章 基于 Ajax 的聊天室 ····· 434
视频讲解：37 分钟
18.1 需求分析 ····· 434
18.2 系统设计 ····· 434
18.2.1 系统目标 ····· 434
18.2.2 系统功能结构 ····· 434
18.2.3 系统流程图 ····· 435
18.3 系统开发及运行环境 ····· 435
18.4 系统文件夹组织结构 ····· 435
18.5 公共类设计 ····· 436
18.5.1 编写并配置字符编码过滤器 ····· 436
18.5.2 编写聊天室相关的 Servlet 实现类 ····· 437
18.6 用户登录模块设计 ····· 438
18.7 聊天室主体功能模块设计 ····· 441
18.7.1 设计聊天室主页面 ····· 441
18.7.2 实时获取并显示在线人员列表 ····· 442
18.7.3 实现用户发言 ····· 443
18.7.4 实时显示聊天内容 ····· 446
18.8 退出聊天室模块 ····· 449
18.8.1 实现安全退出聊天室功能 ····· 450
18.8.2 处理非正常退出聊天室 ····· 451
18.9 开发常见问题与解决方法 ····· 452
18.10 小结 ····· 452
第 19 章 基于 SSH2 的网络商城 ····· 453
视频讲解：49 分钟
19.1 需求分析 ····· 453
19.2 系统设计 ····· 453
19.2.1 系统目标 ····· 453
19.2.2 系统功能结构 ····· 454
19.2.3 系统业务流程图 ····· 454
19.2.4 系统编码规范 ····· 455
19.3 系统开发及运行环境 ····· 455
19.4 数据库与数据表设计 ····· 456

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

19.4.1 E-R 图设计 ····· 456
19.4.2 创建数据库及数据表 ····· 457
19.5 系统文件夹组织结构 ····· 458
19.6 搭建项目环境 ····· 459
19.6.1 配置 Struts 2 ····· 459
19.6.2 配置 Hibernate ····· 462
19.6.3 配置 Spring ····· 463
19.6.4 配置 web.xml ····· 463
19.7 公共类设计 ····· 464
19.7.1 Hibernate 的 Session 初始化类 ····· 464
19.7.2 泛型工具类 ····· 466
19.7.3 数据持久化类 ····· 466
19.7.4 分页设计 ····· 468
19.8 登录注册模块设计 ····· 471
19.8.1 实现注册模块 ····· 472
19.8.2 实现登录模块 ····· 474
19.9 前台商品信息查询模块设计 ····· 475
19.9.1 实现商品分类查询 ····· 476
19.9.2 实现商品搜索 ····· 477
19.9.3 实现前台查询其他商品 ····· 478
19.10 购物车模块设计 ····· 480
19.10.1 实现购物车的基本功能 ····· 480
19.10.2 实现订单的相关功能 ····· 483
19.11 后台商品管理模块设计 ····· 485
19.11.1 实现商品管理功能 ····· 485
19.11.2 实现商品类别管理功能 ····· 489
19.12 后台订单管理模块设计 ····· 492
19.12.1 实现后台订单查询 ····· 492
19.12.2 实现后台订单状态管理 ····· 494
19.13 网站发布 ····· 496
19.13.1 配置 Tomcat 服务器 ····· 496
19.13.2 配置 DNS 服务器 ····· 497
19.14 开发常见问题与解决 ····· 500
19.14.1 页面中出现中文乱码 ····· 500
19.14.2 自定义查询时，当订单号为空会出错 ····· 500
19.15 小结 ····· 501

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第一篇
起步篇

本篇主要内容：

- ❑第 1 章　开启 Java Web 之门
- ❑第 2 章　不可不知的客户端应用技术
- ❑第 3 章　驾驭 Java Web 开发环境
- ❑第 4 章　Java Web 开发必修课之 JSP 语法

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 1 章　开启 Jave Web 之门

视频讲解：19 分钟

作为当前程序开发中最流行编程语言中的一种，Java 语言已经走过了 10 多个春秋，它以面向对象、跨平台、安全、稳定及可扩展等特性吸引了一大批用户。Java 的应用领域比较广，目前的一个主要用途就是开发 Web 应用程序。本章介绍一些 Web 基本知识及 Java Web 的相关内容。

通过学习本章，读者可以达到以下学习目的：

- 认识 Java Web；
- 了解 Web 应用技术；
- 了解 Web 应用程序的工作原理；
- 了解什么是 C/S 和 B/S 结构，以及二者的区别；
- 了解 Java Web 网站的基本构建流程；
- 了解常用的网上资源。

1.1　初识Java Web

1.1.1　Java 概述

Java 是在 1995 年由 Sun 公司推出的一种极富创造力的面向对象的程序设计语言。Sun 研究院的詹姆斯·戈士林主导研发了 Java 语言，并完成了 Java 技术的原始编译器和虚拟机。Java 最初的名字是“Oak”，在 1995 年被重命名为“Java”。

Java 是一种通过解释方式执行的语言，其语法规则和 C++ 类似。它也是一种跨平台的程序设计语言，用其编写的程序可以运行在任何平台和设备上，如 Mac 苹果系统、各种微处理器硬件平台，以及 Windows、UNIX、OS/2 和 Mac OS 等系统平台，真正实现了“一次编写，到处运行”。Java 非常适合于企业网络和 Internet 环境，已成为 Internet 中最有影响力和最受欢迎的编程语言之一。

Java 语言编写的程序既是编译型，又是解释型。程序代码经过编译之后转换为一种称为“Java 字节码”的中间语言，Java 虚拟机 JVM 将解释和运行字节码。编译只运行一次，而解释在每次运行程序时都会进行。编译后的字节码采用一种针对 JVM 优化过的机器码形式保存，虚拟机将字节码解释为机器码，然后在计算机上运行。Java 语言程序代码的编译和运行过程如图 1.1 所示。

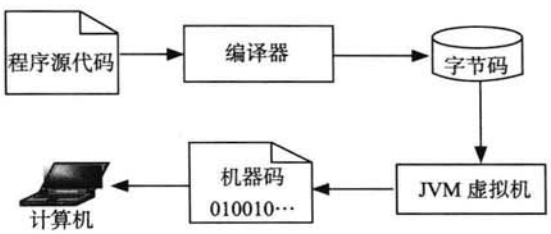


图 1.1　Java 程序代码的编译和运行过程

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

1.1.2　Java 语言的主要特点

Java 语言的主要特点如下。

1．简单

Java 语言的语法简单明了，容易掌握，其简单性主要体现在以下方面。

- 语法规则和 C++ 类似，从某种意义上讲，Java 语言由 C 和 C++ 语言转变而来，所以 C 程序设计人员可以很容易地掌握 Java 语言的语法。
- 简化和提高了 C++，如 Java 使用接口取代了多重继承并取消了指针，因为指针和多重继承通常使程序变得复杂。Java 语言还通过垃圾自动收集大大简化了程序设计人员的资源释放管理工作。

2．面向对象

面向对象是 Java 语言的基础，也是 Java 语言的重要特性，因为其本身就是一种纯面向对象的程序设计语言。Java 提倡万物皆对象，语法规定不能在类外面定义单独的数据和函数。即 Java 语言最外部的数据类型是对象，所有的元素都要通过类和对象来访问。

3．分布性

Java 的分布性包括操作分布和数据分布，操作分布是指在多点不同主机上布置相关操作；数据分布是将数据分别存放在多台不同的主机上，这些主机是网络中的不同成员。Java 可以凭借 URL 对象访问网络对象，访问方式与访问本地系统相同。

4．可移植性

Java 程序具有与体系结构无关的特性，从而使 Java 程序可以方便地移植到网络的不同计算机中；同时 Java 的类库中也实现了针对不同平台的接口，使这些类库也可以移植。

5．安全性

Java 语言删除了类似 C 语言中的指针和内存释放等语法，从而有效地避免了非法操作内存。Java 程序代码要经过代码校验和指针校验等多个测试步骤才能运行，未经允许的 Java 程序不可能出现损害系统平台的行为，而且使用 Java 可以编写防病毒和防修改的系统。

6．健壮性

Java 的设计目标之一是编写各方面均可靠的应用程序，Java 检查程序在编译和运行时的错误，并消除错误。类型检查能帮助用户检查出在开发早期出现的多个错误，很多集成开发工具 IDE（如 Eclipse 和 NetBeans）的出现使编译和运行 Java 程序更加容易。

1.1.3　Java Web 概述

Java Web 是指用 Java 语言来解决相关 Web 领域的技术总和，一个 Web 应用程序包括 Web 客户端和 Web 服务器两个部分，即基于 B/S（浏览器 / 服务器）架构的应用程序。

1．Web 客户端

Web 客户端通常是指用户机上的浏览器，如微软的 IE 浏览器或火狐浏览器等。客户端不需要开发任何用户界面，而统一采用浏览器即可。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2. Web 服务器

Web 服务器是一台或多台可运行 Web 应用程序的计算机，通常我们在浏览器中输入的网站地址，即 Web 服务器的地址。当用户在浏览器的地址栏中输入网站地址并按回车键后，请求即被发送到 Web 服务器。服务器接收到请求后，会返回给用户带有请求资源的响应消息。Java 在服务器端的应用非常丰富，如 Servlet、JSP 和第三方框架等。

Web 客户端和 Web 服务器之间的关系如图 1.2 所示。

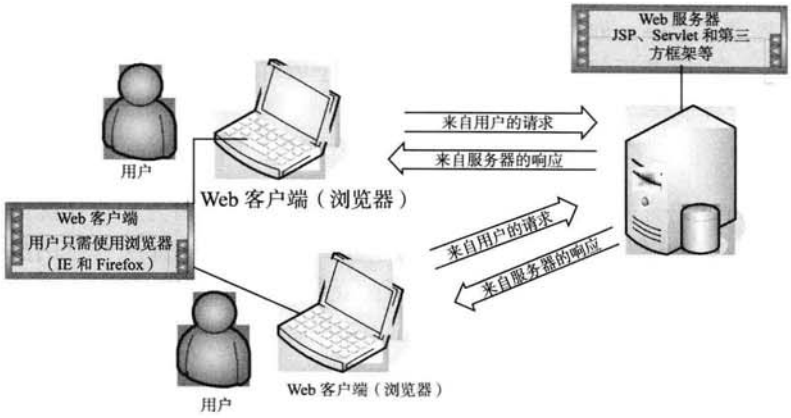


图 1.2　Web 客户端和 Web 服务器之间的关系

1.1.4　Java Web 成功案例

由于 Java 语言具有安全性、健壮性和可移植性等优势，所以应用 Java 开发的 Web 项目应用领域非常广泛。企事业单位（如银行和政府部门）大多采用 Java 开发自己的办公系统。清华大学的本科招生网、工商银行网站、交通银行网站和邮政储蓄银行网站等都是使用 Java Web 开发的，其首页如图 1.3 ～图 1.6 所示。



图 1.3　清华大学招生网首页



图 1.4　工商银行网站首页

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 1.5　交通银行网站首页



图 1.6　邮政储蓄银行网站首页

1.2　Web应用技术

在开发 Web 应用程序时，通常需要应用客户端和服务器两个方面的技术，其中客户端主要用于展现信息内容；服务器端则主要用于处理业务逻辑以及与数据库的交互等。

1.2.1　客户端应用技术

目前比较常用的客户端技术包括 HTML 语言、CSS 样式、Flash 和客户端脚本技术。

1. HTML 语言

HTML 语言是客户端技术的基础，主要用于显示网页信息。它不需要编译，由浏览器解释执行。该语言简单易用，在文件中加入标签，使其可以显示各种各样的字体、图形及闪烁效果。增加了结构和标记，如头元素、文字、列表、表格、表单、框架、图像和多媒体等，并且提供了与 Internet 中其他文档的超链接。例如，在一个 HTML 页中，要应用图像标记插入一张图片，可以使用如图 1.7 所示的 HTML 文件，该 HTML 页运行后的效果如图 1.8 所示。



图 1.7　HTML 文件



图 1.8　运行效果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明 HTML 语言不区分大小写，这一点与 Java 不同，如图 1.7 中的 HTML 标记 <body></body> 标记也可以写为 <BODY></BODY>。

2. CSS

CSS 为样式表（Style Sheet）技术，也称为“层叠样式表”（Cascading Style Sheet）。在制作网页时采用 CSS 样式，可以更加精确地控制页面的布局、字体、颜色、背景和其他效果。只要对相应的代码作一些简单的修改，即可改变整个页面的风格。CSS 大大提高了开发人员对信息展现格式的控制能力，特别是在目前比较流行的 CSS+DIV 布局的网站中，其作用举足轻重。例如，在“心之语许愿墙”网站中，如果删除程序中的 CSS 代码，将显示如图 1.9 所示的效果；添加 CSS 代码后将显示如图 1.10 所示的效果。

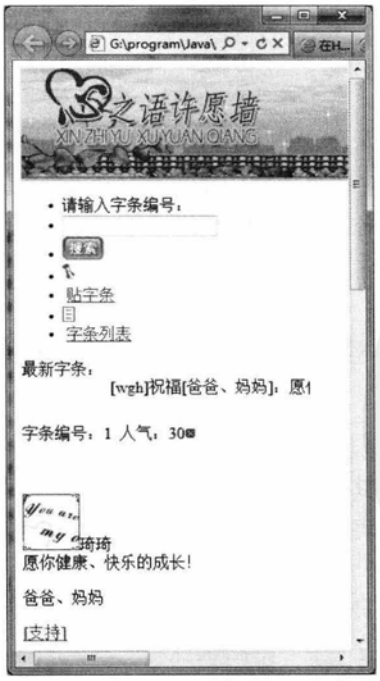


图 1.9 删除 CSS 代码后的效果

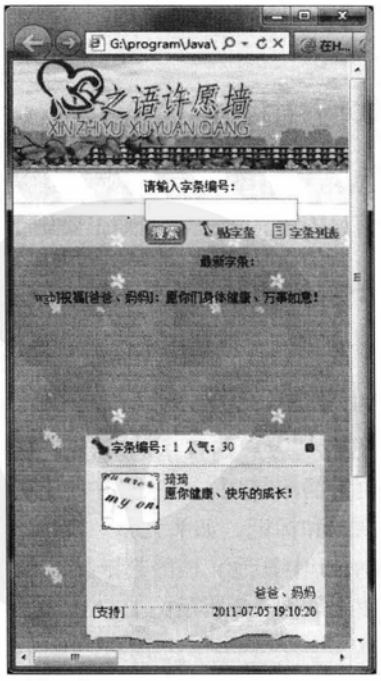


图 1.10 添加 CSS 代码后的效果

技巧 在网页中使用 CSS 样式不仅可以美化页面，而且可以优化网页速度。因为 CSS 样式表文件只是简单的文本格式，不需要安装额外的第三方插件；另外由于 CSS 提供了很多种滤镜效果，所以可以避免使用大量图片。从而大大缩小了文件的大小，提高下载速度。

3. Flash

Flash 是一种交互式矢量动画制作技术，可以包含动画、音频、视频及应用程序。Flash 文件比较小，非常适合在 Web 中应用。目前很多 Web 开发人员都将 Flash 技术引入到网页中，使网页更具有表现力。如应用 Flash 技术实现动态播放网站广告或新闻图片，并且添加随机的转场效果，如图 1.11 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 1.11　在网页中插入的 Flash 动画

4．客户端脚本技术

客户端脚本技术是指嵌入到 Web 页面中的程序代码，这些程序代码是一种解释性的语言，浏览器可以解释客户端脚本。通过脚本语言可以实现以编程的方式控制页面元素，从而增加页面的灵活性。常用的客户端脚本语言有 JavaScript 和 VBScript。

说明　目前应用最为广泛的客户端脚本语言是 JavaScript 脚本，它是 Ajax 的重要组成部分。

1.2.2　服务器端应用技术

目前比较常用的服务器端技术如下。

1．CGI

CGI（Common Gateway Interface，通用网关接口）是最早用来创建动态网页的一种技术，可以使浏览器与服务器之间产生互动关系。它允许使用不同的语言来编写适合的 CGI 程序，然后放在 Web 服务器上运行。当客户端发出请求给服务器时，服务器根据用户请求建立一个新的进程来执行指定的 CGI 程序，并将执行结果以网页形式传输到客户端的浏览器中显示。

2．ASP

ASP（Active Server Page）是一种使用很广泛的开发动态网站的技术，它通过在页面代码中嵌入 VBScript 或 JavaScript 脚本语言来生成动态的内容。在服务器端必须安装适当的解释器后才可以通过调用此解释器来执行脚本程序，然后将执行结果与静态内容部分结合并传送到客户端浏览器中。

3．PHP

PHP（Personal Home Page）是一种开发动态网页的技术，其语法类似于 C，并且混合了 Perl、C++ 和 Java 的一些特性。它是一种开源的 Web 服务器脚本语言，被广泛应用于 UNIX/Linux 平台。由于 PHP 本身的代码对外开放，并且经过软件工程师的检测，因此具有公认的安全性能。

4．ASP.NET

ASP.NET 是一种建立动态 Web 应用程序的技术，是 .NET 框架的一部分，可以使用任何 .NET 兼容的语言来编写 ASP.NET 应用程序。使用 Visual Basic.NET、C#、J# 和 ASP.NET 页面（Web Forms）编译则可以提供比脚本语言更出色的性能。

5．JSP

JSP（Java Server Page）是以 Java 为基础开发的，沿用了 Java 强大的 API 功能。JSP 页面

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

中的 HTML 代码用来显示静态内容部分，嵌入到页面中的 Java 代码与 JSP 标记来生成动态的内容部分。JSP 允许开发人员编写自己的标签库来完成应用程序的特定要求。它可以被预编译，从而提高了程序的运行速度。由于 JSP 开发的应用程序经过一次编译后即可运行，所以在绝大部分系统平台中，代码无须修改即可在支持 JSP 的任何服务器中运行。

1.3 Web应用程序的工作原理

Web 应用程序大体上可以分为静态网站和动态网站，早期的 Web 应用主要是静态页面的浏览，即静态网站。这些网站使用 HTML 语言来编写，放在 Web 服务器上。用户使用浏览器通过 HTTP 协议请求服务器上的 Web 页面，Web 服务器处理接收到的用户请求后发送给客户端浏览器显示给用户，其工作流程如图 1.12 所示。

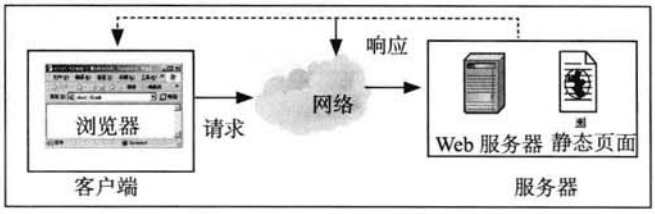


图 1.12 静态网站的工作流程

随着网络的发展，很多线下业务开始向网上发展。基于 Internet 的 Web 应用也变得越来越复杂，用户所访问的资源已不局限于服务器中保存的静态网页。更多的内容需要根据用户的请求动态生成页面信息，即动态网站。这些网站通常使用 HTML 语言和动态脚本语言（如 JSP、ASP 或 PHP 等）编写，并将编写后的程序部署到 Web 服务器中。由 Web 服务器处理动态脚本代码并将其转换为浏览器可以解析的 HTML 代码，最后返回客户端浏览器显示给用户，其工作流程如图 1.13 所示。

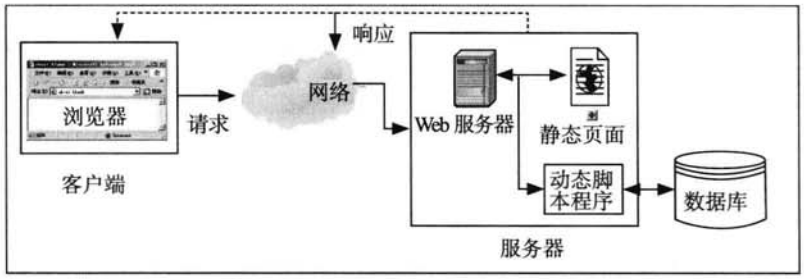


图 1.13 动态网站的工作流程

由此可见，静态网站类似于 10 年前研制的手机。这种手机只能使用出厂时设置的功能和铃声，用户不能进行添加或删除等操作；动态网站则类似于现在研制的手机，用户在使用这些手机时可以根据个人喜好设置。

1.4 网络程序开发体系结构

随着网络技术的不断发展，单机的软件程序难以满足网络计算的需要，各种网络程序开

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

发体系结构应运而生。其中运用最多的网络应用程序开发体系结构可以分为基于客户端 / 服务器的 C/S 结构和基于浏览器 / 服务器的 B/S 结构。

1.4.1　什么是 C/S 结构

在 C/S 结构中服务器通常采用高性能的 PC 或工作站，并采用大型数据库系统（如 Oracle 或 SQL Server）；客户端则需要安装专用的客户端软件，如图 1.14 所示。这种结构可以充分利用两端硬件环境的优势，将任务合理地分配到客户端和服务器，从而降低了系统的通信开销。在 2000 年以前，该结构是网络程序开发领域的主流。

1.4.2　什么是 B/S 结构

在 B/S 结构中客户端不需要开发任何用户界面，而统一采用 IE 或火狐等浏览器。通过 Web 浏览器向 Web 服务器发送请求，由 Web 服务器处理并将处理结果逐级传回客户端，如图 1.15 所示。

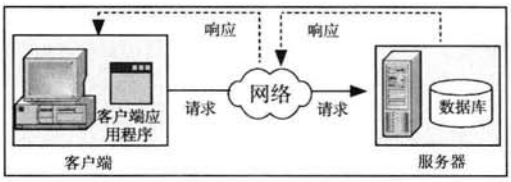


图 1.14　C/S 体系结构

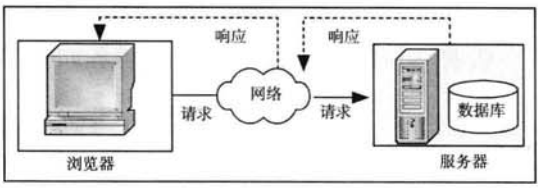


图 1.15　B/S 体系结构

这种结构利用不断成熟和普及的浏览器技术实现原来需要复杂专用软件才能实现的强大功能，从而节约了开发成本。作为一种全新的软件体系结构，其已经成为当今应用软件的首选体系结构。

说明　B/S 由微软公司首次推出，C/S 由 Borland 公司最早研发成功。

1.4.3　B/S 与 C/S 的区别

C/S 和 B/S 结构是当今世界网络程序开发体系结构的两大主流，目前二者都有自己的市场份额和客户群，其优点和缺点如下。

1．开发和维护成本

C/S 结构的开发和维护成本都比 B/S 高，采用这种结构必须为不同客户端开发不同的程序，而且软件的安装、调试和升级均要在所有的客户端上完成。例如，如果一个企业共有 10 个客户端使用一套 C/S 结构的软件，则都需要安装客户端程序。当这套软件即使进行微小的改动，系统维护员都必须卸载客户端原有的软件，然后安装新的版本并配置，维护工作必须进行 10 次。而 B/S 结构的软件则不必在客户端安装及维护，在软件升级后系统维护员只需要将服务器的软件升级到最新版本，所有客户端只要重新登录系统即可使用最新版本的软件。

2．客户端负载

C/S 的客户端不仅负责与用户的交互、收集用户信息，而且还需要通过网络向服务器发送请求以处理数据库、电子表格或文档等。由此可见，应用程序的功能越复杂，客户端程序也就

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

越庞大，这也给软件的维护工作带来了很大的困难。而 B/S 结构的客户端把事务处理逻辑部分交给服务器处理，客户端只需要显示，这样将使应用程序服务器的运行数据负荷较重。一旦发生服务器崩溃等问题，后果不堪设想。因此许多单位都备有数据库存储服务器，以防万一。

3．安全性

C/S 结构适用于专人使用的系统，可以通过严格的管理派发软件达到保证系统安全的目的；B/S 结构的软件由于使用的人数较多，且不固定，相对来说安全性就会低些。

B/S 相对于 C/S 具有更多的优势，现今大量应用程序开始应用 B/S 结构。许多软件公司也争相开发 B/S 版的软件，即 Web 应用程序。随着 Internet 的发展，基于 HTTP 协议和 HTML 标准的 Web 应用呈几何数量级增长，而这些 Web 应用又是由各种 Web 技术开发。

1.5 Java Web网站的基本构建流程

在学习 Java Web 应用程序开发前，需要了解构建一个 Java Web 网站的基本流程。本节将通过一个具体的流程图说明。

构建一个 Java Web 网站的基本流程如图 1.16 所示。

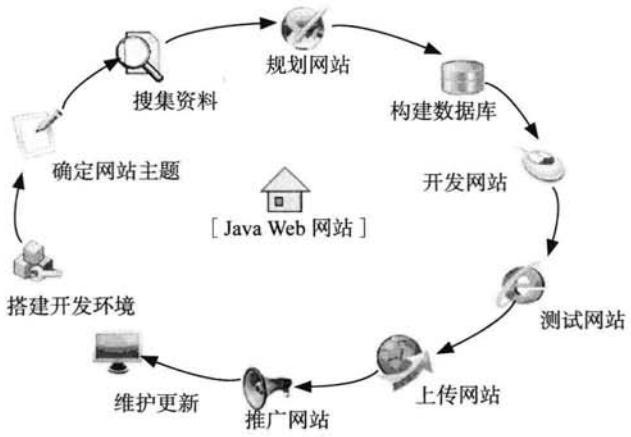


图 1.16　构建一个 Java Web 网站的基本流程

1.6 常用网上资源

为了方便读者学习，本节推荐一些学习 Java Web 应用开发的相关资源，使用这些资源可以帮助找到精通 Java Web 应用开发的捷径。

1.6.1 常用资源下载网

常用的资源下载网站如下。

1）JDK 官方网站：http://www.oracle.com/index.html。

2）Web 服务器 Tomcat 的官方网站：http://tomcat.apache.org。

3）IDE 工具 Eclipse 的官方网站：http://www.eclipse.org。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

4）开源数据库 MySQL 的官方网站：http://www.mysql.com。
5）JSTL 标准标签库的下载网站：http://jstl.java.net/download.html。
6）Struts 官方网站：http://struts.apache.org。
7）Spring 官方网站：http://www.springframework.org。
8）Hibernate 官方网站：http://www.hibernate.org。
9）iBatis 官方网站：http://ibatis.apache.org。

1.6.2　技术社区

为了方便 Java Web 开发人员进行交流与学习，网上提供了多个技术社区。通过登录相关的技术社区，读者可以很好地学习他人的经验技巧，快速提高自己的编程水平。常用的社区如下。

1）CSDN 社区中心：http://community.csdn.net。
2）ITeye 社区：http://www.iteye.com。
3）中文 Java 技术网：http://www.cn-java.com。
4）编程词典服务社区：http://www.mrbccd.com。

1.7　小结

本章首先简单介绍了 Java 和 Java Web、Java Web 开发的成功案例，以及 Web 应用技术，使读者对 Web 应用开发所需的技术有所了解。然后比较了网络程序开发的两种体系结构，并说明了 Web 应用开发所采用的体系结构。最后详细介绍了开发 Java Web 网站的基本流程，以及常用的网上资源。

1.8　习题

填空题

1. Java 是在 1995 年由_____推出的一种极富创造力的_____的程序设计语言。
2. Java 语言的主要特点有简单、_____、分布性、_____、_____和_____。
3. JSP（Java Server Page）是以_____为基础开发的，沿用了_____强大的 API 功能。JSP 页面中的 HTML 代码用来显示_____部分，嵌入到页面中的 Java 代码与 JSP 标记来生成_____部分。

问答题

1. 进行 Java Web 开发时常用的客户端应用技术有哪些?
2. 简述 Web 应用程序的工作原理。
3. 简述什么是 C/S 结构、B/S 结构，以及二者的区别。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 2 章　不可不知的客户端应用技术

视频讲解：113 分钟

在进行 Web 应用开发时，通常情况下，我们会应用 HTML 语言来描述页面中要显示的内容，设置了要显示的内容以后，为了美化页面，还需要 CSS 样式表。另外，有时为了增强页面的灵活性，还需要应用 JavaScript 脚本语言。本章将对 Web 开发时常用的客户端应用技术中的 HTML 5、CSS 样式表和 JavaScript 脚本语言进行详细介绍。

通过学习本章，读者可以达到以下学习目的：

- 了解 HTML 5 的文档结构；
- 掌握 HTML 的文字排版、图片、超链接和表格标记；
- 掌握 HTML 5 新增的语义元素；
- 掌握在 HTML 5 文档中实现音频和视频播放的方法；
- 掌握如何添加 HTML 的表单标记；
- 掌握如何使用 CSS 样式表；
- 了解 JavaScript 脚本语言的基本语法；
- 掌握 JavaScript 脚本语言提供的常用对象；
- 掌握常用的 DOM 技术。

2.1　构建页面内容的HTML 5标记语言

HTML 5 是新一代的 HTML 语言，它将会取代 HTML 4.0 和 XHTML 1.1，成为新一代的 Web 语言。HTML 5 自从 2010 年正式推出以来，就以一种惊人的速度被迅速推广，世界各知名浏览器厂商也对 HTML 5 提供了很好的支持。例如，微软就对 IE 9 做了标准上的改进，使其能够支持 HTML 5。而且 HTML 5 还有一个特点是在老版本的浏览器上也可以正常运行。

2.1.1　了解 HTML 5 文档结构

在介绍 HTML 5 文档结构以前，我们先来看一个基本的 HTML 5 文档，具体代码如图 2.1 所示。

```
1  <!DOCTYPE HTML>                          文档类型
2  <html>                                   根元素
3  <head>
4  <meta charset="utf-8">
5  <title>在HTML页中插入图片</title>          头元素
6  </head>
7                                           主体元素
8  <body>
9  <img src="bccd.jpg" width="400" height="400">
10 </body>
11 </html>
```

图 2.1　一个基本的 HTML 5 文档

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在图 2.1 所示的代码中，第一行代码用于指定文档的类型；第二行和第 11 行为 HTML 5 文档的根元素，也就是 <html> 标记；第 3 行和第 6 行为头元素，也就是 <head> 标记；第 8 行和第 10 行为主体元素，也就是 <body> 标记。

图 2.1 所示代码的运行结果如图 2.2 所示。



图 2.2　一个基本的 HTML 5 文档的运行结果

在对 HTML 5 文档有了一个基本的了解以后，我们再来看一看，组成 HTML 5 文档的各元素。

1. 文档类型

一个标准的 HTML 文档的起始元素为指定文档类型的标记。在 HTML 5 以前的 HTML 文档中，用于指定文档类型的标记代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.
org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

而在 HTML 5 文档中，指定文档类型的代码则更加简短和美观，仅仅使用下面的 15 个字符就可以实现了。

```
<!DOCTYPE HTML>
```

说明　在 HTML 5 文档中，如果你喜欢使用以前版本中提供的指定文档类型的代码，也是可以的。

2. 根元素

HTML 文档的根元素是 <html> 标记。所有 HTML 文件都是以 <html> 标记开头，以 </html> 标记结束。HTML 页面的所有标记都要放置在 <html> 与 </html> 标记中，虽然 <html> 标记并没有实质性的功能，却是 HTML 文件不可缺少的内容。

说明　HTML 标记是不区分大小写的。

3. 头元素

HTML 文件的头元素是 <head> 标记，用于放置 HTML 文档的信息。在 <head> 标记中，可以使用 <title> 标记来指定文档的标题，也可以使用 <meta> 标记来指定字符编码。例如，在 HTML 5 的文档中，我们可以在 <head> 标记中，使用下面的代码的指定字符编码为 UTF-8。

```
<meta charset="utf-8">
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

4. 主体元素

HTML 页面的主体元素为 <body> 标记。<body> 标记也是成对使用的。以 <body> 标记开头，</body> 标记结束。页面中的所有内容都定义在 <body> 标记中。

2.1.2 HTML 文字排版标记

对于 HTML 页面，文字排版标记必不可少，一个美观大方的文字页面能够确切地传达出页面的主要信息。常用的文字排版标记主要包括以下几个。

1. 文字与特殊符号

在 HTML 文档中，要显示普通文字，只需要在 <body> 主体标记中，或者其他子标记中直接输入所需文字就可以了。不过，对于空格和一些特殊符号就不能直接输入了，而是需要通过一个以“&”符号开头，以“;”符号结束的实体名称来代替。常用的特殊符号及其对应的实体名称如表 2.1 所示。

表 2.1 特殊符号及其对应的实体名称

特殊符号	实体名称	特殊符号	实体名称	特殊符号	实体名称
空格		×	×	€	€
“	"	§	§	£	£
&	&	¢	¢	™	™
<	<	¥	¥	©	©
>	>	·	&middot	®	®
+	±				

【例 2.1】 在 HTML 文档上输出版权信息。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.1

```
CopyRight &copy; 2012 www.mrbccd.com 吉林省明日科技有限公司
    本站请使用 IE 9.0 或以上版本 1280&times;1024 为最佳显示效果
```

上面这段代码运行后，将显示如图 2.3 所示的运行结果。

CopyRight © 2012 www.mrbccd.com 吉林省明日科技有限公司
本站请使用IE 9.0或以上版本 1280×1024为最佳显示效果

图 2.3 在 HTML 文档上输出版权符号和空格等文字信息

2. 段落标记

HTML 中的段落标记也是一个很重要的标记，段落标记以 标记开头，以 标记结束。段落标记在段前和段后各添加一个空行，而定义在段落标记中的内容，不受该标记的影响。

3. 换行标记

段落与段落之间是隔行换行的，使得文字的行间距过大；可以使用换行标记
 来完成文字的换行显示。如果直接在 HTML 文档中输入类似于 Word 等文本编辑软件中常用的换行符（〈Enter〉键）是没有用的。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

4．标题标记

在 Word 文档中，可以很轻松地实现不同级别的标题。如果要在 HTML 页面中创建不同级别的标题，可以使用 HTML 语言中的标题标记。在 HTML 标记中，设定了 6 个标题标记，分别为 <h1> 至 <h6>，其中 <h1> 代表 1 级标题，<h2> 代表 2 级标题……<h6> 代表 6 级标题等。数字越小，表示级别越高，文字的字体也就越大。

【例 2.2】 在 HTML 文档上输出 1 ～ 6 级标题，并设置不同的对齐方式。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.2

```
<h1>1 级标题—HTML 5 标记语言 </h1>
<h2>2 级标题—文字排版标记 </h2>
<h3>3 级标题—标题标记 </h3>
<h4 align="left">4 级标题—居左对齐 </h4>
<h5 align="center">5 级标题—居中对齐 </h5>
<h6 align="right">6 级标题—居右对齐 </h6>
```

运行本实例将显示如图 2.4 所示的运行结果。



图 2.4　标题标记

5．文字列表标记

HTML 语言中提供了文字列表标记，文字列表标记可以将文字以列表的形式依次排列。通过这种形式可以更加方便网页的访问者浏览。HTML 中的列表标记主要有无序的列表和有序的列表两种。

（1）无序列表

无序列表是在每个列表项的前面添加一个圆点符号。通过 标记可以创建一组无序列表，其中每一个列表项以 标记表示。

【例 2.3】 使用无序列表在页面中显示文字列表。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.3

```
明日科技编程词典的版本如下：
<br>
<ul>
  <li> 编程词典全能版（98 元）</li>
  <li> 编程词典个人版（298 元）</li>
  <li> 编程词典珍藏版（798 元）</li>
  <li> 编程词典企业版（1998 ～ 6998 元）</li>
  <li> 程序开发资源库（19980 ～ 39980 元）</li>
</ul>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

本实例的运行结果如图 2.5 所示。



图 2.5　在页面中使用无序列表

（2）有序列表

有序列表和无序列表的区别是，使用有序列表标记可以将列表项进行编号。有序列表的标记为 ，每一个列表项前使用 标记。有序列表的列表项是有一定的顺序的。下面将对例 2.3 进行修改，使用有序列表进行编号。

【例 2.4】 使用有序列表对页面中的文字列表进行编号。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.4

```
明日科技编程词典的版本如下：
<br>
<ol>
  <li> 编程词典全能版（98 元）</li>
  <li> 编程词典个人版（298 元）</li>
  <li> 编程词典珍藏版（798 元）</li>
  <li> 编程词典企业版（1998 ～ 6998 元）</li>
  <li> 程序开发资源库（19980 ～ 39980 元）</li>
</ol>
```

运行本实例，结果如图 2.6 所示。



图 2.6　在页面中使用有序列表

2.1.3　图片与超链接标记

在网页中，经常需要插入图片和超链接。在 HTML 页面中，可以使用图片标记来插入图片，使用超链接标记来插入超链接。下面将分别进行介绍。

1. 图片标记

在网页设计时，经常需要插入图片。例如，电子商务网站中对商品进行展示，网络相册中对相片进行展示等。另外，在网页中插入图片也可以起到美化页面的作用。在 HTML 页面中可以使用 标记插入图片。 标记的语法格式如下：

```
<img src="uri" width="value" height="value" border="value" alt=" 提示文字 " >
```

 标记的属性说明如表 2.2 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 2.2　超链接标记的属性说明

属　性	说　明
src	用于指定图片的来源
width	用于指定图片的宽度
height	用于指定图片的高度
border	用于指定图片外边框的宽度，默认值为 0
alt	用于指定当图片无法显示时显示的文字

例如，在 HTML 文档中，插入一个图片标记用于显示一幅图片，关键代码如下：

```
<img src="mrlogo.jpg">
```

运行上面的代码，将显示如图 2.7 所示的图片。



图 2.7　应用 标记显示一张图片

2．超链接标记

超链接是网页页面中最重要的元素之一。一个网站是由多个页面组成的，页面之间根据链接确定相互的导航关系。单击网页上的链接文字或者图像后，就可以跳转到另一个网页。每一个网页都有唯一的地址，在英文中被称作 URL（Uniform Resource Locator，统一资源定位符）。在 HTML 文档中，使用 <a> 标记来定义超链接。超链接标记的基本语法格式如下：

```
<a href="url" hreflang="language" name="bookmarkName" type="mimeType"charset="code"
shape="area" coords="coordinate " target="target" tabindex="value"accesskey="key">
Linkcontent
</a>
```

属性说明如表 2.3 所示。

表 2.3　超链接标记的属性说明

属　性	说　明
href	用于指定超链接地址，可以是绝对路径（需要提供完全的路径，包括适用的协议，如 http 或 ftp 等），也可以是相对路径（只要属于同一网站之下就可以，可以不在同一个目录下）
hreflang	用于指定超链接位置所使用的语言
name	用于指定超链接的标识名
type	用于指定超链接位置所使用的 MIME 类型
charset	用于指定超链接位置所使用的编码方式
target	用于指定超链接的目标窗口，可选值如表 2.4 所示

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（续）

属　性	说　明
tabindex	用于指定按下〈Tab〉键时移动的顺序，从属性值最小的开始移动，其取值范围为 0 ~ 32 767
linkcontent	用于指定设置超链接的内容，可以是文字，也可以是图片
accesskey	用于为超链接设置快捷键

表 2.4　链接的目标窗口属性

属 性 值	说　明
_parent	在上一级窗口中打开。一般使用框架页时经常使用
_blank	在新窗口中打开
_self	在同一个窗口中打开，这项一般不用设置
_top	在浏览器的整个窗口中打开，忽略任何框架

技巧 IE 浏览器中，按住〈Alt〉键，再按下 accesskey 属性定义的快捷键（焦点将移动到该超链接），再按回车键即可执行该超链接；在火狐浏览器中，按住〈Alt+Shift〉键，再按下 accesskey 属性定义的快捷键即可执行该超链接。

例如，在 HTML 文档中，添加一个链接到明日图书网的超链接的代码如下：

```
<a href="http://www.mingribook.com"> 明日图书网 </a>
```

2.1.4　HTML 5 新增的语义元素

在 HTML 5 中，为了使文档的结构更加清晰明确，追加了几个与页眉、页脚、内容区块等文档结构相关联的语义元素，下面将分别进行介绍。

1. <header> 元素

<header> 元素表示页面中一个内容区域或整个页面的标题。通常情况下，它可能是一个页面中（指主体标记中）的第一个元素，可以包含站点的标题、Logo 和旗帜广告等。

【例 2.5】 应用 <header> 标记定义页面的页眉，包括网站的 Logo 和标题。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.5

```
<header>
<img src="mrlogo.jpg">
     <h1> 吉林省明日科技有限公司 </h1>
</header>
```

上面这段代码运行后，将显示如图 2.8 所示的运行结果。

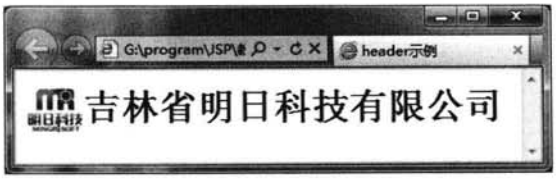


图 2.8　应用 <header> 标记定义的页眉

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2．<footer> 元素

<footer> 元素表示整个页面或页面中一个内容区域块的脚注。脚注中通常包含一些基本信息，例如，日期、作者、相关文档的链接或版权信息等。尽管脚注通常情况下都是放置在页面或者内容区块的最底部，但是它并不是必须放置在最底部，也可以根据实际需要进行合理的放置。

【例 2.6】 应用 <footer> 标记定义页面的脚注，这里为显示版权信息。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.6

```
<footer>
    <ul>
        <li>CopyRight &copy; 2012 www.mrbccd.com 吉林省明日科技有限公司 </li>
        <li>
        本站请使用 IE 9.0 或以上版本 1280&times;1024 为最佳显示效果
        </li>
    </ul>
</footer>
```

运行上面的代码，将显示如图 2.9 所示的运行结果。



图 2.9　应用 <footer> 标记定义页面的脚注

3．<section> 元素

<section> 元素表示页面中的一个区域。例如，章节、页眉、页脚或页面中的其他部分。可以与 h1、h2、h3、h4 等元素结合起来使用，用于标识文档结构。

【例 2.7】 应用 <section> 标记在页面中定义一个区域。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.7

```
<section>
    <h2>section 标记的使用 </h2>
    <p> 编程词典系列软件是为各类爱好编程者和各级程序开发人员提供了
    学、查、用为一体的数字化编程软件。</p>
    <footer>2012 年 7 月 12 日 </footer>
</section>
```

上面这段代码相当于在 HTML 4 中使用 标记在页面中定义一个区域，运行结果如图 2.10 所示。

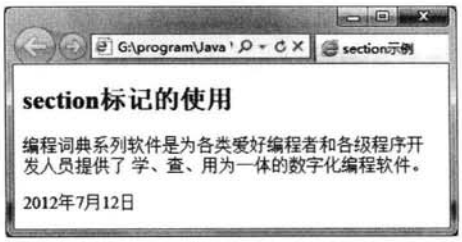


图 2.10　应用 <section> 标记定义一个区域

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

4. <article> 元素

<article> 元素代表文档、页面或应用程序中的所有“正文”部分，它所描述的内容应该是独立的、完整的、可以独自被外部引用的，可以是一篇博文、报刊中的一篇文章、一篇论坛帖子、一段用户评论或任何独立于上下文中其他部分的内容。除了内容部分，一个 <article> 元素通常有自己的标题和脚注等内容。

说明 <article> 标记的内容独立于文档的其余部分。

【例 2.8】 应用 <article> 元素在页面中定义一篇文章。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.8

```
<article>
   <header>
       <h1> 苹果美容 </h1>
    </header>
    <p> 苹果素有 " 水果之王 " 的美称，它含有丰富的维生素 C，能让皮肤细嫩、柔滑而白皙。苹果面膜的做法很简单，将苹果去皮去核切块后放入搅拌机中搅成泥状，干性皮肤的美眉在苹果泥中加入新鲜的牛奶或蜂蜜，油性皮肤的美眉则可加入少量蛋清，搅拌均匀后涂在脸上，敷 10~15 分钟后洗净，你会发现肤色有明显变化哦。</p>
    <footer>
      <p>2012-7-12</p>
    </footer>
</article>
```

5. <aside> 元素

<aside> 元素用来表示当前页面或文章的附属信息部分。可以包含与当前页面或主要内容相关的引用、侧边栏、广告、导航条等信息。

【例 2.9】 应用 <aside> 元素定义页面侧栏。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.9

```
<aside>
  <h2> 侧栏 </h2>
  <ul>
    <li><a href="http://www.mingribook.com"> 明日图书 </a> </li>
    <li><a href="http://www.mingrisoft.com"> 明日软件 </a></li>
    <li><a href="http://www.mrbccd.com"> 编程词典 </a></li>
  </ul>
 </aside>
```

本实例的运行结果如图 2.11 所示。



图 2.11 应用 <aside> 标记定义侧栏

6. <nav> 元素

<nav> 元素用来表示页面中的导航链接区域，其中包括一个页面（例如，一篇文章顶端的一个目录，它可以链接到同一页面的锚点）或一个站点内的链接。但是，并不是链接的每一个集合都是一个 nav，只需要将主要的、基本的链接组放进 nav 元素即可。例如，在页脚中通常会有一组链接，包括服务条款、版权声明、联系方式等。对于这些 footer 元素就足够放置了。一个页面中可以拥有多个 nav 元素，作为页面整体或不同部分的导航。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【例 2.10】 应用 <nav> 元素定义页面的主导航栏。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.10

```
<nav>
  <ul>
    <li><a href="#"> 首页 </a></li>
    <li><a href="#"> 新品上市 </a></li>
    <li><a href="#"> 特价商品 </a></li>
    <li><a href="#"> 畅销商品 </a></li>
    <li><a href="#"> 购物车 </a></li>
    <li><a href="#"> 查看订单 </a></li>
    <li><a href="#"> 会员资料修改 </a></li>
  </ul>
</nav>
```

本实例的运行结果如图 2.12 所示。

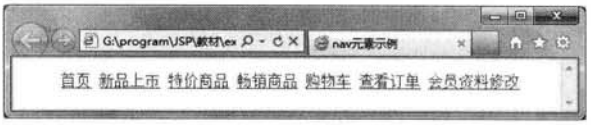


图 2.12　应用 <nav> 标记定义导航区域

2.1.5　制作表格

表格是网页中十分重要的组成元素，常用来存储数据。表格通常由标题、表头、行和单元格组成。在 HTML 语言中，表格使用 <table> 标记来定义。不过定义表格时，只使用 <table> 标记是不够的，还需要定义表格中的行、列、标题等内容。在 HTML 页面中定义表格，需要使用以下几个标记。

（1）表格标记 <table>

<table>…</table> 标记表示整个表格。<table> 标记中有很多属性，例如 width 属性用来设置表格的宽度，border 属性用来设置表格的边框，align 属性用来设置表格的对齐方式，bgcolor 属性用来设置表格的背景色等。

（2）标题标记 <caption>

标题标记以 <caption> 开头，以 </caption> 结束，标题标记也有一些属性，例如 align 和 valign 等。

（3）表头标记 <th>

表头标记以 <th> 开头，以 </th> 结束，也可以通过 align、background、colspan、valign 等属性来设置表头。

（4）表格行标记 <tr>

表格行标记以 <tr> 开头，以 </tr> 结束，一组 <tr> 标记表示表格中的一行。<tr> 标记要嵌套在 <table> 标记中使用，该标记也具有 align、background 等属性。

（5）单元格标记 <td>

单元格标记 <td> 又称为列标记，一个 <tr> 标记中可以嵌套若干个 <td> 标记。该标记也具有 align、background、valign 等属性。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【例 2.11】 在 HTML 文档中定义学生成绩表。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.11

```
<table width="300" height="150" border="1" align="center">
  <caption> 学生考试成绩单 </caption>
  <tr>
    <td align="center" valign="middle"> 姓名 </td>
    <td align="center" valign="middle"> 语文 </td>
    <td align="center" valign="middle"> 数学 </td>
    <td align="center" valign="middle"> 英语 </td>
  </tr>
  <tr>
    <td align="center" valign="middle"> 琦琦 </td>
    <td align="center" valign="middle">89</td>
    <td align="center" valign="middle">92</td>
    <td align="center" valign="middle">97</td>
  </tr>
  <tr>
    <td align="center" valign="middle"> 宁宁 </td>
    <td align="center" valign="middle">93</td>
    <td align="center" valign="middle">86</td>
    <td align="center" valign="middle">80</td>
  </tr>
  <tr>
    <td align="center" valign="middle"> 婷婷 </td>
    <td align="center" valign="middle">85</td>
    <td align="center" valign="middle">86</td>
    <td align="center" valign="middle">90</td>
  </tr>
</table>
```

运行本实例，将显示如图 2.13 所示的运行结果。

学生考试成绩单

姓名	语文	数学	英语
琦琦	89	92	97
宁宁	93	86	80
婷婷	85	86	90

图 2.13 在 HTML 文档中显示学生成绩单

2.1.6 播放音频和视频

在 HTML 5 出现以前，如果开发者想要在 Web 页面中包含视频，必须使用 <object> 和 <embed> 元素，而这两个元素使用起来需要指定很多参数，比较麻烦。现在 HTML 5 提供了两个用来播放音频和视频的标记 <audio> 和 <video>，使用起来比较简单。下面将对这两个标记进行介绍。

说明 到目前为止，还不是所有浏览器都支持 <audio> 和 <video> 标记，不过在新版本的浏览器中将对该标记提供支持。其中，IE 9、Firefox 3.5、Safari 3.2、Chrome 3.0 和 Opera 10.5 浏览器都已经开始支持 <audio> 和 <video> 标记了。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（1）播放音频标记 <audio>

<audio> 标记专门用来播放音频数据。它的使用方法比较简单，例如，要播放网络中的一首 MP3 音乐，那么可以使用下面的代码。

```
<audio src="http://www.mingrisoft.com/temp/cuckoo.mp3" autoplay>您的浏览器不支持
&lt;audio&gt; 标记！ </audio>
```

说明　<audio> 标记可以支持多种音频格式，包括 Ogg、MP3、AAC 和 WAV 等，不同浏览器支持的音频格式也不尽相同。例如，IE 9 支持 MP3 和 ACC；Firefox 3.6+ 支持 Ogg 和 WAV；Chrome 10+ 支持 Ogg、MP3、AAC 和 WAV；Opera 11+ 支持 Ogg 和 WAV。

由于各个浏览器支持的音频格式不尽相同，所以应用 <audio> 标记在页面中播放音频时，需要根据不同的浏览器提供不同格式的音频文件，这样才能让要播放的音频数据在不同的浏览器上都能播放。例如，要播放一首萨克斯曲《茉莉花》，那么，我们需要使用下面的代码。

```
<audio autoplay>您的浏览器不支持 &lt;audio&gt; 标记！
    <source src="jasmine.ogg" type="audio/ogg">
    <source src="jasmine.mp3" type="audio/mpeg">
</audio>
```

这样就可以做到，在 IE 9 浏览器中能播放这首乐曲，而在 Firefox 3.6+ 中也能播放。

（2）播放视频标记 <video>

<video> 标记用于播放视频数据。它的语法格式如下：

```
<video src="url" width="value" height="value" autoplay="true|false"
controls="true|false" >
您的浏览器不支持 &lt;video&gt; 标记！
</video>
```

<video> 标记的属性说明如表 2.5 所示。

表 2.5　<video> 标记的属性说明

属　性	说　明
src	用于指定要播放的视频，它的属性值为视频的 URL 地址
width	用于指定播放器的宽度
height	用于指定播放器的高度
autoplay	用于指定是否自动播放视频，属性值为 true 或 false。为 true 时表示自动播放，否则为不自动播放
controls	用于指定是否显示播放控制组件，属性值为 true 或 false。为 true 时表示显示播放控制组件，否则为不显示播放控制组件

【例 2.12】 在 HTML 文档中播放 MP4 视频。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.12

```
<video src="mingrisoft.mp4" autoplay="true" controls="true" >
   您的浏览器不支持 &lt;video&gt; 标记！
</video>
```

运行本实例，将显示如图 2.14 所示的运行结果。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 2.14　在 IE 9 中播放 MP4 视频

说明　<video> 标记可以支持多种视频格式，包括 Ogg、MP4、WebM 等，不同浏览器支持的视频格式也不尽相同。例如，IE 9 支持 MP4；Firefox 3.6+ 支持 Ogg；Chrome 11+ 支持 Ogg、MP4、WebM 和 WAV；Opera 11+ 支持 Ogg 和 WebM。

由于各个浏览器支持的视频格式不尽相同，所以当应用 <video> 标记在页面中播放视频时，需要根据不同的浏览器提供不同格式的视频文件，这样才能让要播放的视频数据在不同的浏览器上都能播放。例如，要播放一个宣传视频，那么，我们需要使用下面的代码。

```
<video width="720" height="576" autoplay="true" controls="true" >
您的浏览器不支持 &lt;video&gt; 标记!
<source src="mingrisoft.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'/>
<source src="Big.ogv" type='video/ogg; codecs="theora, vorbis"' />
</video>
```

这样就可以做到，在 IE 9 浏览器中能播放这段视频，而在 Firefox 3.6+ 中也能播放这段视频了。

2.1.7　表单标记

通过 HTML 表单，可以将用户输入的信息提交到服务器中，经服务器处理后，再回传给客户端的浏览器。从而实现网站与用户之间的交互。所以说 HTML 表单是进行动态网站开发必不可少的内容。下面将对 HTML 中的表单标记进行介绍。

1. <from> 标记

<form> 标记用于在页面中插入表单，在该标记中可以定义处理表单数据程序的 URL 地址等信息。<form> 标记的语法格式如下：

```
<form name="form name" method="method" action="url" enctype="value"
target="target_win">
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
...
</form>
```

- name 属性：用于指定表单的名称。
- method 属性：用于指定表单的提交方式，其可选项包括 POST 和 GET。
- action 属性：用于指定表单提交的 URL 地址（相对地址和绝对地址），也就是表单的处理页。
- enctype 属性：用于设置表单内容的编码方式，其可选值包括以下 3 个。
 - text/plain：以纯文本形式传送信息。
 - application/x-www-form-urlencoded：默认的编码形式。
 - multipart/form-data：使用 MINE 编码。
- target 属性：用于指定返回信息的显示方式，其可选值如表 2.6 所示。

表 2.6　target 属性的可选值

值	说　明
_blank	将返回信息显示在新的窗口中
_parent	将返回信息显示在父级窗口中
_self	将返回信息显示在当前窗口中
_top	将返回信息显示在顶级窗口中

例如，在 HTML 文档中插入一个表单标记，设置表单名称为 form，当用户提交表单时，提交至 action.html 页面进行处理。

```
<form id="form1" name="form" method="post" action="action.html" target="_blank">
</form>
```

2. <input> 表单输入标记

表单输入标记是使用最频繁的表单标记，通过这个标记可以向页面中添加单行文本、多行文本、按钮等。<input> 标记的语法格式如下：

```
<input type="image" disabled="disabled" checked="checked" width="digit"height=
"digit" maxlength="digit" readonly="" size="digit" src="uri" usemap="uri" alt=""
name="checkbox" value="checkbox">
```

<input> 标记的属性如表 2.7 所示。

表 2.7　<input> 标记的属性

属 性 值	说　明
type	用于指定添加的是哪种类型的输入字段，共有 17 个可选值，如表 2.8 所示
disabled	用于指定输入字段不可用，即字段变成灰色。其属性值可以为空值，也可以指定为 disabled
checked	用于指定输入字段是否处于被选中状态，用于 type 属性值为 radio 和 checkbox 的情况下。其属性值可以为空值，也可以指定为 checked
width	用于指定输入字段的宽度，用于 type 属性值为 image 的情况下
height	用于指定输入字段的高度，用于 type 属性值为 image 的情况下

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（续）

属 性 值	说 明
maxlength	用于指定输入字段可输入文字的个数，用于 type 属性值为 text 和 password 的情况下，默认没有字数限制
readonly	用于指定输入字段是否为只读。其属性值可以为空值，也可以指定为 readonly
size	用于指定输入字段的宽度，当 type 属性为 text 和 password 时，以文字个数为单位，当 type 属性为其他值时，以像素为单位
src	用于指定图片的来源，只有当 type 属性为 image 时有效
usemap	为图片设置热点地图，只有当 type 属性为 image 时有效。属性值为 URI，URI 格式为“#+<map> 标记的 name 属性值”。例如，<map> 标记的 name 属性值为 Map，该 URI 为 #Map
alt	用于指定当图片无法显示时显示的文字，只有当 type 属性为 image 时有效
name	用于指定输入字段的名称
value	用于指定输入字段默认数据值，当 type 属性为 checkbox 和 radio 时，不可省略此属性，为其他值时，可以省略。当 type 属性为 button、reset 和 submit 时，指定的是按钮上的显示文字；当 type 属性为 checkbox 和 radio 时，指定的是数据项选定时的值

type 属性是 <input> 标记中非常重要的内容，决定了输入数据的类型。该属性值的可选项如表 2.8 所示。

表 2.8 type 属性的属性值

属 性 值	说 明	示 例	属 性 值	说 明	示 例
text	文本框	请输入用户名	submit	提交按钮	提交
password	密码域	•••••••	reset	重置按钮	重置
file	文件域	"H:\图片\apple.jpg" 选择...	button	普通按钮	按钮
url	* URL 地址		hidden	隐藏域	
email	* E-mail 地址		image	图像域	登录
color	* 颜色选择器	#000000 其他·····	radio	单选按钮	男 女
datetime	* 日期时间选择器		* date	日期选择器	
number	* 数字选择器	6	checkbox	复选按钮	音乐 美术
range	* 滑块				

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明	在表 2.8 中所列出的属性值中，“说明”栏目中被标记为 * 号的属性值为 HTML 5 新增加的功能，在 Opera 11 浏览器中，可以看到表 2.8 所列的示例效果。

说明	HTML 5 还提供了两个有用的属性，即 placeholder（占位文本）属性和 autofocus（自动聚焦）属性。使用 placeholder 属性，可以为输入框设置占位文本；使用 autofocus 属性时，当页面载入后，该输入框将自动获得焦点。不过，到目前为止，IE 9 还不支持这两个属性。

说明	当输入框为空且失去焦点时，占位文本会显示出来，而一旦用户输入了实际内容，或者该输入框获得了焦点，这个占位文本就会消失。

3．<select>…</select> 下拉列表标记

<select> 标记可以在页面中创建下拉列表，此时的下拉列表是一个空的列表，要使用 <option> 标记向列表中添加内容。<select> 标记的语法格式如下：

```
<select name="name" size="digit" multiple="multiple" disabled="disabled">
</select>
```

- name 属性：用于指定列表框的名称。
- size 属性：用于指定列表框中显示的选项数量，超出该数量的选项可以通过拖动滚动条查看。
- disabled 属性：用于指定当前列表框不可使用（变成灰色）。
- multiple 属性：用于让多行列表框支持多选。

例如，在 HTML 文档中插入一个下拉列表标记，设置名称为 zone，并为其设置相应的菜单选项。

```
<select name="zone" >
    <option value="1"> 吉林省 </option>
    <option value="2"> 辽宁省 </option>
    <option value="3"> 黑龙江省 </option>
    <option value="4"> 河北省 </option>
    <option value="5"> 河南省 </option>
    <option value="6"> 山西省 </option>
</select>
```

运行上面这段代码，将显示如图 2.15 所示的运行结果。



图 2.15　下拉列表框

4．<textarea> 多行文本标记

<textarea> 为多行文本标记，与单行文本相比，多行文本可以输入更多的内容。通常情况下，<textarea> 标记出现在 <form> 标记的标记内容中。<textarea> 标记的语法格式如下：

```
<textarea cols="digit" rows="digit" name="name" disabled="disabled"
readonly="readonly" wrap="value"> 默认值 </textarea>
```

<textarea> 标记的属性说明如表 2.9 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 2.9 <textarea> 标记的属性说明

属　性	说　明
name	用于指定多行文本框的名称，当表单提交后，在服务器端获取表单数据时应用
cols	用于指定多行文本框显示的列数（宽度）
rows	用于指定多行文本框显示的行数（高度）
disabled	用于指定当前多行文本框不可使用（变成灰色）
readonly	用于指定当前多行文本框为只读
wrap	用于设置多行文本中的文字是否自动换行，可选值有 hard（默认值，表示自动换行，如果文字超过 cols 属性所指的列数就自动换行，并且在提交到服务器时，换行符同时被提交）、soft（表示自动换行，如果文字超过 cols 属性所指的列数就自动换行，但提交到服务器时，换行符不被提交）和 off（表示不自动换行，如果想让文字换行，只能按下〈Enter〉键强制换行）

例如，在表单中添加一个编辑框，名称为 content，5 行 30 列，文字换行方式为 hard，具体代码如下：

```
<textarea name="content" cols="30" rows="5" wrap="hard">? </textarea>
```

运行上面的代码，将显示如图 2.16 所示的运行结果。

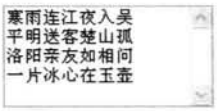


图 2.16 编辑框

2.2 美化页面的CSS样式表

CSS 是 W3C 协会为弥补 HTML 在显示属性设定上的不足而制定的一套扩展样式标准，它的全称是“Cascading Style Sheet”。CSS 标准中重新定义了 HTML 中原来的文字显示样式，增加了一些新概念，如类、层等，可以对文字进行重叠、定位等。在 CSS 还没有引入到页面设计之前，传统的 HTML 语言要实现页面美化在设计上是十分麻烦的，例如要设计页面中文字的样式，如果使用传统的 HTML 语句来设计页面就不得不在每个需要设计的文字上都定义样式。CSS 的出现改变了这一传统模式。

2.2.1 样式表的定义与引用

CSS 样式表有 4 种定义与引用方式，分别为行内样式、内嵌式、链接式和导入式，我们可以根据实际情况选择合适的定义方式。

1. 行内样式

行内样式是比较直接的一种样式，直接定义在 HTML 标记之内，通过 style 属性来实现。这种方式也比较容易令初学者接受，但是灵活性不强。

例如，要在 HTML 文档中定义一个段落，并为其使用行内样式设置该段落中的文字颜色为蓝色，可以使用下面的代码。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<p style="color:blue">文字</p>
```

2. 内嵌式

内嵌式样式表就是在页面中使用<style></style>标记将CSS样式包含在页面中。例如，在HTML文档中，定义一段设置段落文字颜色的内嵌样式表，可以使用下面的代码。

```
<style>
   p{
         color:red;      /*设置文字颜色*/
   }
</style>
```

说明 内嵌式样式表的形式没有行内标记表现得直接，但是能够使页面更加规整。

与行内样式相比，内嵌式样式表更加便于维护，但是如果每个网站都不可能由一个页面构成，而每个页面中相同的HTML标记都要求有相同的样式，此时使用内嵌式样式表就显得比较笨重，可以用链接式样式表解决这一问题。

3. 链接式

链接外部CSS样式表是最常用的一种引用样式表的方式，将CSS样式定义在一个单独的文件中，然后在HTML页面中通过<link>标记引用。这是一种最为有效的使用CSS样式表的方式。

<link>标记的语法结构如下：

```
<link rel="stylesheet" href="path" type="text/css">
```

- rel：定义外部文档和调用文档间的关系。
- href：CSS文档的绝对或相对路径。
- type：指的是外部文件的MIME类型。

例如，我们已经创建了一个名称为mystyle.css的外部样式表文件，在HTML文档中，就可以通过下面的代码来引用该外部样式表文件。

```
<link rel="stylesheet" href="mystyle.css" type="text/css"/>
```

4. 导入式

导入式是指在内部样式表的<style>标记里导入一个外部样式表，导入时使用@import实现。例如，已经创建了一个名称为mystyle.css的外部样式表文件，要将该样式表文件导入到当前的内部样式表中，可以使用下面的代码。

```
<style>
   @import "mystyle.css";
</style>
```

说明 当一个HTML文档中同时存在以上4种形式定义的样式时，它们的优先级从高到低依次是行内样式、内嵌式或链接式、导入式。其中，当内嵌式与链接式同时存在时，因为内嵌式的优先级比链接式的优先级高，所以优先显示内嵌式。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2.2.2 CSS 规则

在 CSS 样式表中包括 3 部分内容：选择符、属性和属性值。语法格式为：

```
选择符 { 属性 : 属性值 ; }
```

- 选择符：又称选择器，是 CSS 中很重要的概念，所有 HTML 语言中的标记都是通过不同的 CSS 选择器进行控制的。本章将在 2.2.3 节中详细向大家介绍。
- 属性：主要包括字体属性、文本属性、背景属性、布局属性、边界属性、列表项目属性、表格属性等内容。其中一些属性只有部分浏览器支持，因此使 CSS 属性的使用变得更加复杂。
- 属性值：为某属性的有效值。属性与属性值之间以“:”号分隔。当有多个属性时，使用“;”分隔。图 2.17 为大家标注了 CSS 语法中的选择器、属性与属性值。

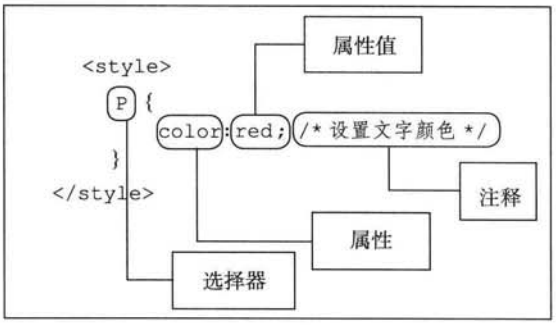


图 2.17 CSS 语法

2.2.3 CSS 选择器

CSS 选择器常用的是标记选择器、类别选择器和 ID 选择器等。使用选择器即可对不同的 HTML 标签进行控制，来实现各种效果。下面对各种选择器进行详细介绍。

1. 标记选择器

大家知道 HTML 页面是由很多标记组成，例如图片标记 、超链接标记 <a>、表格标记 <table> 等。而 CSS 标记选择器就是声明页面中哪些标记采用哪些 CSS 样式。例如 a 选择器，就是用于声明页面中所有 <a> 标记的样式风格。

例如，定义 a 标记选择器，在该标记选择器中定义超链接的字体大小与颜色。

```
<style>
   a{
          font-size:9pt;
          color:red;
   }
</style>
```

说明 在 CSS 样式中，使用起始标记“/*”和结束标记“*/”来定义注释。例如，要添加注释信息“设置文字大小”，可以使用代码“/* 设置文字大小 */”。

2. 类别选择器

使用标记选择器非常快捷，但是会有一定的局限性，如果在页面中声明标记选择器，那

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

么页面中所有使用该标记的内容会有相应的变化。假如页面中有 3 个 <h2> 标记，如果想要每个 <h2> 的显示效果都不一样，使用标记选择器就无法实现了，这时就需要引入类别选择器。

类别选择器的名称由用户自己定义，并以“.”开头，定义的属性与属性值也要遵循 CSS 规范。要应用类别选择器的 HTML 标记，只需使用 class 属性来声明即可。

【例 2.13】 使用类别选择器控制页面中字体的样式。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.13

```
<head>
<meta charset="utf-8">
<title> 类别选择器示例 </title>
<!-- 以下为定义的 CSS 样式 -->
<style>
   .one{                                          /* 定义类名为 one 的类别选择器 */
          font-family: 宋体 ;                     /* 设置字体 */
          font-size:24px;                         /* 设置字体大小 */
          color:red;                              /* 设置字体颜色 */
     }
   .two{
          font-family: 宋体 ;                     /* 设置字体 */
          font-size:18px;                         /* 设置字体大小 */
          color:red;                              /* 设置字体颜色 */
     }
   .three{
          font-family: 宋体 ;                     /* 设置字体 */
          font-size:12px;                         /* 设置字体大小 */
          color:red;                              /* 设置字体颜色 */
     }
</style>
</head>
<body>
   <h2 class="one"> 应用了选择器 one </h2> <!-- 定义样式后页面会自动加载样式 -->
   <p> 正文内容 1 </p>
    <h2 class="two"> 应用了选择器 two</h2>
   <p> 正文内容 2 </p>
   <h2 class="three"> 应用了选择器 three </h2>
   <p> 正文内容 3 </p>
</body>
```

在上面的代码中，页面中的第一个 <h2> 标记应用了 one 选择器，第二个 <h2> 标记应用了 two 选择器，第 3 个 <h2> 标记应用了 three 选择器，运行结果如图 2.18 所示。



图 2.18　应用类别选择器控制页面文字样式

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明 在 HTML 标记中，不仅可以应用一种类别选择器，也可以应用多种类别选择器，这样可使 HTML 标记同时加载多个类别选择器的样式。在使用多种类别选择器时，两个类别选择器之间使用空格进行分割，例如“<h2 class="size color">”。

3. ID 选择器

ID 选择器是通过 HTML 页面中的 ID 属性来进行选择以增添样式，与类别选择器的基本使用方法相同，但需要注意的是由于 HTML 页面中不能包含两个相同的 id 标记，因此定义的 ID 选择器也就只能被使用一次。

命名 ID 选择器要以“#”开始，后加 HTML 标记中的 ID 属性值。

【例 2.14】 使用 ID 选择器控制页面中字体的样式。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.14

```
<style>                    <!-- 定义 id 选择器 -->
#one{
     font-size:18px;
}
#two{
     font-size:24px;
}
#three{
     font-size:36px;
}
</style>
<body>
<p id="one">ID 选择器 </p>                          <!-- 在页面定义标记，则自动应用样式 -->
<p id="two">ID 选择器 2</p>
<p id="three">ID 选择器 3</p>
</body>
```

运行本段代码，结果如图 2.19 所示。



图 2.19 使用 ID 选择器控制页面文字大小

2.2.4 CSS 常用属性

通过 CSS 可以美化页面，这当然离不开 CSS 提供的常用属性。下面将通过一个表格给出 Web 页面设计常用的一些 CSS 属性。CSS 常用属性如表 2.10 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 2.10　CSS 常用属性

属性名称	功 能 描 述
background	用于设置背景颜色、背景图片（包括排列方式、位置、是否固定等）。该属性可指定多个属性值，各属性值以空格分隔，没有先后顺序
border	用于设置边框的宽度、边框的样式和边框的颜色。该属性可指定多个属性值，各属性值以空格分隔，没有先后顺序
clear	用于指定不允许有浮动对象的边，通常用于清除浮动效果
color	用于指定文本的颜色
display	用于指定对象是否显示及如何显示，也就是设置对象的显示形式
float	用于指定对象是否浮动及如何浮动。在默认情况下，一个块级元素在水平方向会自动伸展，直到包含它的元素的边界。而在竖直方向和兄弟元素依次排列的情况下不能并排，这时就可以应用 float 属性进行设置，使其可以并排显示
font	用于设置字体样式、字体粗细、文字的大小、行高和文字的字体等
height	用于指定对象的高度
list-style	用于指定项目符号的种类、指定图片作为项目符号和项目符号排列的位置
margin	用于指定对象的外边距，也就是对象与对象之间的距离。该属性可指定 1 至 4 个属性值，各属性值以空格分隔
padding	用于指定对象的内边距，也就是对象的内容与对象边框之间的距离。该属性可指定 1 至 4 个属性值，各属性值以空格分隔
text-align	用于指定文本的对齐方式
text-decoration	用于指定文本的显示样式，其属性值有 line-through（删除线）、overline（上划线）、underline（下划线）、blink（闪烁效果，Firefox 和 Opera 可以看到效果）和 none（无效果）等
position	用于指定对象的定位方式
width	用于指定对象的宽度

【例 2.15】 为一个段落标记添加内嵌式 CSS 样式，设置它的外边框和内边距。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.15

```
<style>
   p{
          border:1px #000 solid;                    /* 设置 1 像素的黑色实线的外边框 */
          padding:5px;                              /* 设置内边距为 5 像素 */
   }
</style>
```

如果在定义该内嵌式 CSS 样式的 HTML 文档上定义一个 <p> 标记，那么将显示如图 2.20 所示的运行结果。

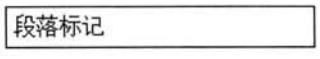


图 2.20　为段落标记设置外边框和内边距

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2.3 客户端交互的JavaScript

JavaScript 是 Web 页面中的一种脚本编程语言，它是基于对象和事件驱动并具有较高安全性能的解释型语言。JavaScript 可以直接嵌入在 HTTP 页面中，把静态页面转变成支持用户交互并响应应用事件的动态页面。在 JSP 程序中，经常应用 JavaScript 进行数据验证、控制浏览器以及生成时钟、日历和时间戳文档等。

2.3.1 JavaScript 的语言基础

JavaScript 与其他语言一样，有其自己的基本语法、数据类型、运算符和流程控制语句，这些都是学习 JavaScript 的语言基础。下面将对 JavaScript 的语言基础进行详细介绍。

1. 基本语法

JavaScript 的语法比较简单，在编写时，只需要注意以下事项就可以了。

- JavaScript 区分大小写。例如，变量 goodsName 与变量 goodsname 是两个不同的变量。
- 每行结尾可以加分号表示语句结束，也可以不加分号。如果不加结尾处的分号，JavaScript 会自动将该行代码的结尾作为语句的结尾。
- 变量是弱类型的，因此在定义变量时，只使用 var 运算符就可以将变量初始化为任意的值。
- 使用大括号标记代码段，被封装在大括号内的语句将按顺序执行。
- JavaScript 提供了两种注释，即使用双斜线“//”开头的单行注释和以“/*”开头，以“*/”结尾的多行注释。

2. 数据类型

JavaScript 有 6 种数据类型，如表 2.11 所示。

表 2.11 JavaScript 提供的数据类型

类 型	含 义		说 明	示 例
int	数值	整型	整数，可以为正数、负数或 0	17，-80，0
float		浮点型	浮点数，可以使用实数的普通形式或科学记数法表示	3.141 592 7，6.16e4
string	字符串类型		字符串，是用单引号或双引号括起来的一个或多个字符	'qq'，"一片冰心在玉壶"
boolean	布尔型		只有 true 或 false 两个值	true，false
object	对象类型			
null	空类型		没有任何值	
undefined	未定义类型		指变量被创建，但未被赋值	

3. 变量

变量是指程序中一个已经命名的存储单元，其主要作用是为数据操作提供存放信息的容器。在 JavaScript 中，可以使用命令 var 声明变量，语法格式如下：

```
var number;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在声明变量的同时也可以对变量进行赋值：

```
var number=100;
```

由于 JavaScript 采用弱类型的形式，所以在声明变量时，不需要指定变量的类型，而变量的类型将根据其变量赋值来确定。例如：

```
var number=17;                                    // 数值型
var str="爱护地球";                               // 字符型
```

但是变量命名必须遵循以下规则。

- 必须以字母或下划线开头，中间可以是数字、字母或下划线，但是不能有空格或加号、减号等符号。

说明　虽然 JavaScript 的变量可以任意命名，但是在实际编程时，最好使用便于记忆且有意义的变量名称，以增加程序的可读性。

- 不能使用 JavaScript 中的关键字。JavaScript 的关键字如表 2.12 所示。

表 2.12　JavaScript 的关键字

关 键 字	关 键 字	关 键 字	关 键 字	关 键 字	关 键 字
abstract	continue	finally	instanceof	private	this
boolean	default	float	int	public	throw
break	do	for	interface	return	typeof
byte	double	function	long	short	true
case	else	goto	native	static	var
catch	extends	implements	new	super	void
char	false	import	null	switch	while
class	final	in	package	synchronized	with

注意　关键字同样不可用做函数名、对象名及自定义的方法名等。

4．运算符

在 JavaScript 中提供了赋值运算符、算术运算符、关系运算符、逻辑运算符、条件运算符和字符串运算符 6 种常用的运算符。下面进行详细介绍。

（1）赋值运算符

最基本的赋值运算符是等于号“=”，用于对变量进行赋值，而其他运算符可以和赋值运算符“=”联合使用，构成组合赋值运算符。JavaScript 支持的常用赋值运算符如表 2.13 所示。

表 2.13　JavaScript 常用的赋值运算符

运 算 符	描　述	示　例
=	将右边表达式的值赋给左边的变量	username="mr"
+=	将运算符左边的变量加上右边表达式的值赋给左边的变量	a+=b　// 相当于 a=a+b
-=	将运算符左边的变量减去右边表达式的值赋给左边的变量	a-=b　// 相当于 a=a-b
=	将运算符左边的变量乘以右边表达式的值赋给左边的变量	a=b　// 相当于 a=a*b
/=	将运算符左边的变量除以右边表达式的值赋给左边的变量	a/=b　// 相当于 a=a/b

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（续）

运 算 符	描 述	示 例
%=	将运算符左边的变量用右边表达式的值求模，并将结果赋给左边的变量	a%=b // 相当于 a=a%b
&=	将运算符左边的变量与右边表达式的值进行逻辑与运算，并将结果赋给左边的变量	a&=b // 相当于 a=a&b
\|=	将运算符左边的变量与右边表达式的值进行逻辑或运算，并将结果赋给左边的变量	a\|=b // 相当于 a=a\|b
^=	将运算符左边的变量与右边表达式的值进行异或运算，并将结果赋给左边的变量	a^=b // 相当于 a=a^b

（2）算术运算符

算术运算符等同于数学运算，即在程序中进行加、减、乘、除等运算。在 JavaScript 中常用的算术运算符如表 2.14 所示。

表 2.14 JavaScript 常用的算术运算符

运 算 符	描 述	示 例
+	加运算符	1+7 // 返回值为 8
−	减运算符	9−2 // 返回值为 7
*	乘运算符	4*9 // 返回值为 36
/	除运算符	6/3 // 返回值为 2
%	求模运算符	9%4 // 返回值为 1
++	自增运算符。该运算符有两种情况：i++（在使用 i 之后，使 i 的值加 1）；++i（在使用 i 之前，先使 i 的值加 1）	i=1; j=i++ //j 的值为 1，i 的值为 2 i=1; j=++i //j 的值为 2，i 的值为 2
−−	自减运算符。该运算符有两种情况：i−−（在使用 i 之后，使 i 的值减 1）；−−i（在使用 i 之前，先使 i 的值减 1）	i=7; j=i−− //j 的值为 7，i 的值为 6 i=7; j=−−i //j 的值为 6，i 的值为 6

注意 执行除法运算时，0 不能做除数。如果 0 做除数，返回结果则为 Infinity。

【例 2.16】 编写 JavaScript 代码，应用算术运算符计算商品金额。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.16

```
<script language="javascript">
    var price=197;                          // 定义商品单价
    var number=10;                          // 定义商品数量
    var sum=price*number;                   // 计算商品金额
    alert(sum);                             // 显示商品金额
</script>
```

运行结果如图 2.21 所示。



图 2.21 弹出对话框显示商品金额

（3）关系运算符

关系运算符的基本操作过程是：首先对操作数进行比较，这个操作数可以是数字也可以是字符串，然后返回一个布尔值 true 或 false。JavaScript 支持的常用关系运算符如表 2.15 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 2.15　JavaScript 常用的关系运算符

运 算 符	描　述	示　例
<	小于	1<6　// 返回值为 true
>	大于	7>10　// 返回值为 false
<=	小于等于	10<=10　// 返回值为 true
>=	大于等于	3>=6　// 返回值为 false
==	等于。只根据表面值进行判断，不涉及数据类型	"17"==17　// 返回值为 true
===	绝对等于。根据表面值和数据类型同时进行判断	"17"===17　// 返回值为 false
!=	不等于。只根据表面值进行判断，不涉及数据类型	"17"!=17　// 返回值为 false

（4）逻辑运算符

逻辑运算符返回一个布尔值，通常和比较运算符一起使用，用来表示复杂的比较运算，常用于 if、while 和 for 语句中。JavaScript 中常用的逻辑运算符如表 2.16 所示。

表 2.16　JavaScript 常用的逻辑运算符

运 算 符	描　述	示　例
!	逻辑非。否定条件，即！假＝真，！真＝假	!true　// 值为 false
&&	逻辑与。只有当两个操作数的值都为 true 时，值才为 true	true && flase　// 值为 false
\|\|	逻辑或。只要两个操作数其中之一为 true，值就为 true	true \|\| false　// 值为 true

（5）条件运算符

条件运算符是 JavaScript 支持的一种特殊的 3 目运算符，同 Java 中的 3 目运算符类似，其语法格式如下：

```
操作数？结果 1: 结果 2
```

如果“操作数”的值为 true，则整个表达式的结果为“结果 1”，否则为“结果 2”。

（6）字符串运算符

字符串运算符是用于两个字符型数据之间的运算符，除了比较运算符外，还可以是“+”和“+=”运算符。其中，“+”运算符用于连接两个字符串，而“+=”运算符则连接两个字符串，并将结果赋给第一个字符串。

【例 2.17】 在网页中弹出一个提示对话框，显示进行字符串运算后变量 a 的值。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.17

```
var a="One "+"world ";                    // 将两个字符串连接后的值赋值给变量 a
a+="One Dream"                            // 连接两个字符串，并将结果赋给第一个字符串
alert(a);
```

上述代码的执行结果如图 2.22 所示。



图 2.22　弹出提示对话框

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2.3.2 JavaScript 的流程控制语句

流程控制语句对于任何一门编程语言都是至关重要的，JavaScript 也不例外。在 JavaScript 中提供了 if 条件判断语句、for 循环语句、while 循环语句、do... while 循环语句、break 语句、continue 语句和 switch 多分支语句 7 种流程控制语句。

1. if 条件判断语句

对变量或表达式进行判定并根据判定结果进行相应的处理，可以使用 if 语句。if 语句的语法格式如下：

```
if(条件表达式){
   语句序列 1    //条件满足时执行
}else{
   语句序列 2    //条件不满足时执行
}
```

执行上述 if 语句时，首先计算“条件表达式（任意的逻辑表达式）”的值，如果为 true，就执行“语句序列 1”，执行完毕后结束该 if 语句；否则执行“语句序列 2”，执行后同样结束该 if 语句。

说明 上述 if 语句是典型的二路分支结构。其中 else 部分可以省略，而且“语句序列”为单一语句时，其两边的大括号可以省略。

2. for 循环语句

for 语句是 JavaScript 语言中应用比较广泛的条件语句。通常，for 语句使用一个变量作为计数器来执行循环的次数，这个变量就称为循环变量。for 语句的语法格式如下：

```
for(循环变量赋初值；循环条件；循环变量增值){
  循环体；
}
```

- 循环变量赋初值：是一条初始化语句，用来对循环变量进行初始化赋值。
- 循环条件：是一个包含比较运算符的表达式，用来限定循环变量的上限。如果循环变量超过了该上限，则停止该循环语句的执行。
- 循环变量增值：用来指定循环变量的步幅。

for 语句可以使用 break 语句来中止循环语句的执行。break 语句默认情况下是终止当前的循环语句。

3. while 循环语句

while 语句是另一种基本的循环语句，其结构和 for 语句有些类似，但是 while 语句不包含循环变量的初始化及循环变量的步幅。其语法格式如下：

```
while (条件表达式){
   循环体
}
```

使用 while 语句时，必须先声明循环变量并且在循环体中指定循环变量的步幅，否则 while 语句将成为一个死循环。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

4．Do...while 循环语句

Do...while 语句和 while 语句非常相似，所不同的是它是在循环底部检测循环表达式，而不是像 while 语句那样在循环顶部进行检测。这就保证了循环体至少被执行一次。Do...while 语句的语法格式如下：

```
do{
 循环体
} while (条件表达式);
```

【例 2.18】 分别利用 for、while 和 do...while 循环语句将数字 9 格式化为 00009，并输出到页面上的代码如下。

实例位置：光盘\MR\源码\第 2 章\2.18

```
var str="9";
for(i=0;i<4;i++ ){
     str="0"+str;
}
document.write(str+" ");
```

```
var i=0;
var str="9";
while(i<4){
     str="0"+str;
     i++;
}
document.write(str+" ");
```

```
var i=0;
var str="9";
do{
     str="0"+str;
     i++;
} while(i<4);
document.write(str);
```

运行结果如图 2.23 所示。

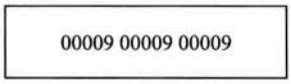


图 2.23　在页面中输出格式化后的结果

5．break 语句

break 语句用于退出包含在最内层的循环或者退出一个 switch 语句。break 语句的语法格式如下：

```
break;
```

说明 break 语句通常用在 for、while、do...while 或 switch 语句中。

6．continue 语句

continue 语句和 break 语句类似，所不同的是，continue 语句用于中止本次循环，并开始下一次循环。其语法格式如下：

```
continue;
```

说明 continue 语句只能应用在 while、for 或 do...while 语句中。

7．switch 语句

switch 是典型的多路分支语句，其作用与嵌套使用 if 语句基本相同，但 switch 语句比 if 语句更具有可读性，而且 switch 语句允许在找不到一个匹配条件的情况下执行默认的一组语句。switch 语句的语法格式如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
switch (expression){
    case judgement1:
          statement1;
          break;
    case judgement2:
          statement2;
          break;
      ...
    default:
         defaultstatement;
      break;
}
```

- expression：为任意的表达式或变量。
- judgement：为任意的常数表达式。当 expression 的值与某个 judgement 的值相等时，就执行此case后的statement语句，如果expression的值与所有的judgement的值都不相等时，则执行 default 后面的 defaultstatement 语句。
- break：用于结束 switch 语句，从而使 JavaScript 只执行匹配的分支。如果没有了 break 语句，则该 switch 语句的所有分支都将被执行，switch 语句也就失去了使用的意义。

【例 2.19】 应用 switch 语句输出今天是星期几。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.19

```
<script language="javascript">
var now=new Date();                          //获取系统日期
var day=now.getDay();                        //获取星期
var week;
switch (day){
    case 1:
        week="星期一";
     break;
case 2:
        week="星期二";
     break;
case 3:
        week="星期三";
     break;
case 4:
        week="星期四";
     break;
case 5:
        week="星期五";
     break;
case 6:
        week="星期六";
     break;
default:
        week="星期日";
      break;
}
document.write("今天是"+week);               //输出中文的星期
</script>
```

程序的运行结果如图 2.24 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 2.24　实例运行结果

2.3.3　JavaScript 函数的定义及调用

在 JavaScript 中，函数可以分为定义和调用两部分。下面分别进行介绍。

1．函数的定义

在 JavaScript 中，定义函数最常用的方法是通过 function 语句实现，其语法格式如下：

```
function functionName([parameter1, parameter2,...]){
   statements;
   [return expression;]
}
```

- functionName：必选，用于指定函数名。在同一个页面中，函数名必须是唯一的，并且区分大小写。
- parameter1, parameter2,…… ：可选，用于指定参数列表。当使用多个参数时，参数间使用逗号进行分隔。一个函数最多可以有 255 个参数。
- statements：必选，是函数体，用于实现函数功能的语句。
- return expression;：可选，用于返回函数值。expression 为任意的表达式、变量或常量。

2．函数的调用

函数的调用比较简单，如果要调用不带参数的函数，则使用函数名加上括号即可；如果要调用的函数带参数，则在括号中加上需要传递的参数，如果包含多个参数，各参数间用逗号分隔。

如果函数有返回值，那么可以使用赋值语句将函数值赋给一个变量。

说明　在 JavaScript 中，由于函数名区分大小写，所以，在调用函数时，也需要注意函数名的大小写。

2.3.4　事件处理

JavaScript 可以以事件驱动的方式直接对客户端的输入作出响应，无须经过服务器端程序；也就是说，JavaScript 是事件驱动的。它可以使在图形界面环境下的一切操作变得简单化。下面将对事件及事件处理程序进行详细介绍。

1．什么是事件处理程序

JavaScript 与 Web 页面之间的交互是通过用户操作浏览器页面时触发相关事件来实现的。例如，在页面载入完毕时将触发 onload（载入）事件、当用户单击按钮时将触发按钮的 onclick

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

事件等。事件处理程序则是用于响应某个事件而执行的处理程序。事件处理程序可以是任意 JavaScript 语句，但通常使用特定的自定义函数（function）来对事件进行处理。

2. 事件类型

多数浏览器内部对象都拥有很多事件，下面将以表格的形式给出常用的事件及何时触发这些事件。JavaScript 的常用事件如表 2.17 所示。

表 2.17 JavaScript 的常用事件

事　件	何时触发
onabort	对象载入被中断时触发
onblur	元素或窗口本身失去焦点时触发
onchange	改变 <select> 元素中的选项或在焦点后内容发生改变时触发
onclick	单击鼠标左键时触发。当光标的焦点在按钮上，并按下回车键时，也会触发该事件
ondblclick	双击鼠标左键时触发
onerror	出现错误时触发
onfocus	任何元素或窗口本身获得焦点时触发
onkeydown	键盘上的按键（包括〈Shift〉或〈Alt〉等键）被按下时触发，如果一直按着某键，则会不断触发。当返回 false 时，取消默认动作
onkeypress	键盘上的按键被按下，并产生一个字符时发生。也就是说，当按下 <Shift> 或 <Alt> 等键时不触发。如果一直按下某键时，会不断触发。当返回 false 时，取消默认动作
onkeyup	释放键盘上的按键时触发
onload	页面完全载入后，在 Window 对象上触发；所有框架都载入后，在框架集上触发； 标记指定的图像完全载入后，在其上触发；或 <object> 标记指定的对象完全载入后，在其上触发
onmousedown	单击任何一个鼠标按键时触发
onmousemove	鼠标指针在某个元素上移动时持续触发
onmouseout	将鼠标指针从指定的元素上移开时触发
onmouseover	鼠标指针移到某个元素上时触发
onmouseup	释放任意一个鼠标按键时触发
onreset	单击重置按钮时，在 <form> 上触发
onresize	当窗口或框架的大小发生改变时触发
onscroll	在任何带滚动条的元素或窗口上滚动时触发
onselect	选中文本时触发
onsubmit	单击提交按钮时，在 <form> 上触发
onunload	页面完全卸载后，在 Window 对象上触发；或者所有框架都卸载后，在框架集上触发

3. 事件处理程序的调用

在使用事件处理程序对页面进行操作时，最主要的是如何通过对象的事件来指定事件处理程序。指定方式主要有以下两种。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（1）在 JavaScript 中的处理

在 JavaScript 中分配事件处理程序，首先需要获得要处理对象的引用，然后将要执行的处理函数赋值给对应的事件处理程序。例如：

```
<img src="images/download.GIF" id="img_download">
<script language="javascript">
var img=document.getElementById("img_download");
img.onclick=function(){
 alert("单击了图片");
}
</script>
```

在页面中加入上面的代码，并运行，当单击图片 img_download 时，将弹出“单击了图片”对话框。

注意　在 JavaScript 中分配事件处理程序时，事件处理程序名称必须小写，才能正确响应事件。

（2）在 HTML 中的处理

在 HTML 中分配事件处理程序，只需要在 HTML 标记中添加相应的事件处理程序的属性，并在其中指定作为属性值的代码或是函数名称即可。例如：

```
<img src="images/download.GIF" onClick="alert('您单击了图片');">
```

在页面中加入上面的代码，并运行，当单击图片 img_download 时，将弹出“您单击了图片”对话框。

2.3.5　常用对象

JavaScript 提供了一些内部对象，下面将介绍最常用的 String、Date 和 window 3 种对象。

1．String 对象

String 对象是动态对象，需要创建对象实例后才能引用它的属性和方法。在创建一个 String 对象变量时，可以使用 new 运算符与来创建，也可以直接将字符串赋给变量。例如 strValue="hello" 与 strVal=new String("hello") 是等价的。String 对象的常用属性和方法如表 2.18 所示。

表 2.18　String 对象的常用属性和方法

属性 / 方法	说　明
length	用于返回 String 对象的长度
split(separator,limit)	用 separator 分隔符将字符串划分成子串并将其存储到数组中，如果指定了 limit，则数组限定为 limit 给定的数，separator 分隔符可以是多个字符或一个正则表达式，它不作为任何数组元素的一部分返回
substr(start,length)	返回字符串中从 start 开始的 length 个字符的子字符串
substring(from,to)	返回以 from 开始、以 to 结束的子字符串
replace(searchValue,replaceValue)	将 searchValue 换成 replaceValue 并返回结果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（续）

属性 / 方法	说　明
charAt(index)	返回字符串对象中的指定索引号的字符组成的字符串，位置的有效值为 0 到字符串长度减 1 的数值。一个字符串的第一个字符的索引位置为 0，第二个字符位于索引位置 1，依此类推。当指定的索引位置超出有效范围时，charAt 方法返回一个空字符串
toLowerCase()	返回一个字符串，该字符串中的所有字母都被转换为小写字母
toUpperCase()	返回一个字符串，该字符串中的所有字母都被转换为大写字母

说明 由于在 JavaScript 中可以将用单引号或双引号括起来的一个字符串当做一个字符串对象的实例，所以可以直接在某个字符串后面加上点“.”去调用 String 对象的属性和方法。

2．Date 对象

Date 对象是一个有关日期和时间的对象。它具有动态性，即必须使用 new 运算符创建一个实例。例如：

```
mydate = new Date();
```

Date 对象没有提供直接访问的属性，只具有获取和设置日期和时间的方法。Date 对象的方法如表 2.19 所示。

表 2.19　Date 对象的方法

获取日期和时间的方法	说　明	设置日期和时间的方法	说　明
getFullYear()	返回用 4 位数表示的年份	setFullYear()	设置年份，用 4 位数表示
getMonth()	返回月份（0 ~ 11）	setMonth()	设置月份（0 ~ 11）
getDate()	返回日数（1 ~ 31）	setDate()	设置日数（1 ~ 31）
getDay()	返回星期（0 ~ 6）	setDay()	设置星期（0 ~ 6）
getHours()	返回小时数（0 ~ 23）	setHours()	设置小时数（0 ~ 23）
getMinutes()	返回分钟数（0 ~ 59）	setMinutes()	设置分钟数（0 ~ 59）
getSeconds()	返回秒数（0 ~ 59）	setSeconds()	设置秒数（0 ~ 59）
getTime()	返回 Date 对象的内部毫秒表示	setTime()	使用毫秒形式设置 Date 对象

3．window 对象

window 对象是浏览器（网页）的文档对象模型结构中最高级的对象，它处于对象层次的顶端，提供了用于控制浏览器窗口的属性和方法。由于 window 对象使用十分频繁，又是其他对象的父对象，所以在使用 window 对象的属性和方法时，JavaScript 允许省略 window 对象的名称。

window 对象的常用属性如表 2.20 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 2.20　window 对象的常用属性

属　性	描　述
frames	表示当前窗口中所有 frame 对象的集合
location	用于代表窗口或框架的 Location 对象。如果将一个 URL 赋予给该属性，那浏览器将加载并显示该 URL 指定的文档
length	窗口或框架包含的框架个数
history	对窗口或框架的 History 对象的只读引用
name	用于存放窗口的名字
status	一个可读写的字符，用于指定状态栏中的当前信息
parent	表示包含当前窗口的父窗口
opener	表示打开当前窗口的父窗口
closed	一个只读的布尔值，表示当前窗口是否关闭。当浏览器窗口关闭时，表示该窗口的 window 对象并不会消失，不过它的 closed 属性被设置为 true

window 对象的常用方法如表 2.21 所示。

表 2.21　window 对象的常用方法

方　法	描　述
alert()	弹出一个警告对话框
confirm()	显示一个确认对话框，单击“确认”按钮时返回 true，否则返回 false
prompt()	弹出一个提示对话框，并要求输入一个简单的字符串
close()	关闭窗口
focus()	把键盘的焦点赋予给顶层浏览器窗口。在多数平台上，这将使窗口移到最前边
open()	打开一个新窗口
setTimeout(timer)	在经过指定的时间后执行代码
clearTimeout()	取消对指定代码的延迟执行
resizeBy(offsetx,offsety)	按照指定的位移量设置窗口的大小
print()	相当于单击浏览器工具栏中的“打印”按钮
setInterval()	周期执行指定的代码
clearInterval()	停止周期性地执行代码

【例 2.20】 通过按钮打开一个新窗口，并在新窗口的状态栏中显示当前年份。

实例位置：光盘 \MR\ 源码 \ 第 2 章 \2.20

Step 1 在主窗口中应用以下代码添加一个用于打开一个新窗口的按钮：

```
<input name="button" value="打开新窗口" type="button"
onclick="window.open('newWindow.html','','width=360,height=100,status=yes')">
```

Step 2 创建一个新的 JSP 文件，名称为 newWindow.jsp，在该文件中添加以下用于在状态栏中显示当前年份的代码。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<script language="javascript">
var mydate=new Date();                                    //创建当前时间所对应的日期时间对象
window.status="现在是："+mydate.getFullYear()+"年！";
                                                          //设置状态栏显示文本
</script>
```

运行结果如图 2.25 所示。



图 2.25　运行结果

2.3.6　DOM 技术

DOM 是 Document Object Model（文档对象模型）的简称，是表示文档（如 HTML 文档）和访问、操作构成文档的各种元素（如 HTML 标记和文本串）的应用程序接口（API）。它提供了文档中独立元素的结构化、面向对象的表示方法，并允许通过对象的属性和方法访问这些对象。另外，文档对象模型还提供了添加和删除文档对象的方法，这样能够创建动态的文档内容。DOM 也提供了处理事件的接口，它允许捕获和响应用户以及浏览器的动作。下面将对其进行详细介绍。

1．DOM 的分层结构

在 DOM 中，文档的层次结构以树形表示。树是倒立的，树根在上，枝叶在下，树的节点表示文档中的内容。DOM 树的根节点是个 Document 对象，该对象的 documentElement 属性引用表示文档根元素的 Element 对象。对于 HTML 文档，表示文档根元素的 Element 对象是 <html> 标记，<head> 和 <body> 元素是树的枝干。下面以一个简单的 HTML 文档说明 DOM 的分层结构。

```
<html>
<head>
    <title>一个 HTML 文档</title>
</head>
<body>
    欢迎访问明日科技网站！
    <br>
    <a href="http://www.mingribook.com"> http://www.mingribook.com</a>
 </body>
</html>
```

上面的 HTML 文档对应的 Document 对象的层次结构如图 2.26 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

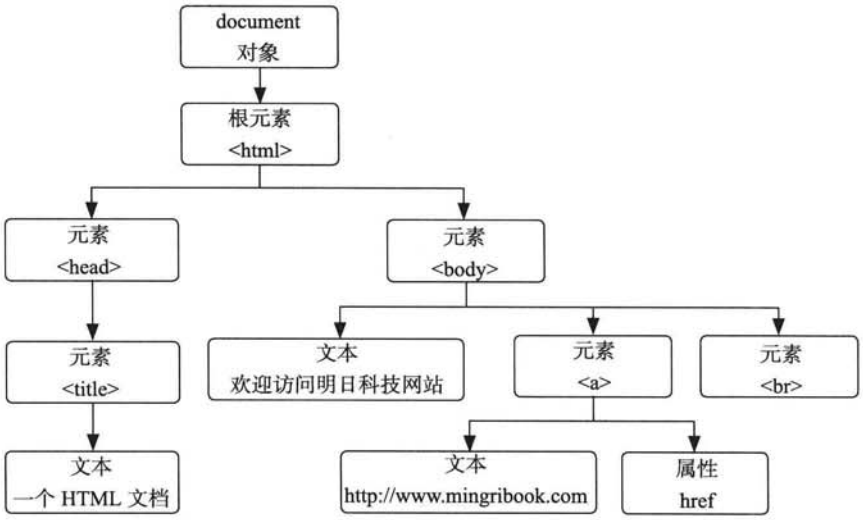


图 2.26　Document 对象的文档层次结构图

说明　在树形结构中，直接位于一个节点之下的节点被称为该节点的子节点（children）；直接位于一个节点之上的节点被称为该节点的父节点（parent）；位于同一层次，具有相同父节点的节点是兄弟节点（sibling）；一个节点的下一个层次的节点集合是该节点的后代（descendant）；一个节点的父节点、祖父节点及其他所有位于它之上的节点都是该节点的祖先（ancestor）。

2．遍历文档

在 DOM 中，HTML 文档各个节点被视为各种类型的 Node 对象，并且将 HTML 文档表示为 Node 对象的树。对于任何一个树形结构来说，最常做的就是遍历树。在 DOM 中，可以通过 Node 对象的 parentNode、firstChild、nextChild、lastChild、previousSibling 等属性来遍历文档树。Node 对象的常用属性如表 2.22 所示。

表 2.22　Node 对象的属性

属　性	类　型	描　述
parentNode	Node	节点的父节点，没有父节点时为 null
childNodes	NodeList	节点的所有子节点的 NodeList
firstChild	Node	节点的第一个子节点，没有则为 null
lastChild	Node	节点的最后一个子节点，没有则为 null
previousSibling	Node	节点的上一个节点，没有则为 null
nextChild	Node	节点的下一个节点，没有则为 null
nodeName	String	节点名
nodeValue	String	节点值
nodeType	short	表示节点类型的整型常量（表 10.23）

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

由于 HTML 文档的复杂性，DOM 定义了 nodeType 来表示节点的类型。下面以列表的形式给出 Node 对象的节点类型、节点名、节点值及节点类型常量，如表 2.23 所示。

表 2.23 Node 对象的节点类型、节点名、节点值及节点类型常量

节点类型	节 点 名	节 点 值	节点类型常量
Attr	属性名	属性值	ATTRIBUTE_NODE（2）
CDATASection	#cdata-section	CDATA 段内容	CDATA_SECTION_NODE（4）
Comment	#comment	注释的内容	COMMENT_NODE（8）
Document	#document	null	DOCUMENT_NODE（9）
DocumentFragment	#document-fragment	null	DOCUMENT_FRAGMENT_NODE（11）
DocumentType	文档类型名	null	DOCUMENT_TYPE_NODE（10）
Element	标记名	null	ELEMENT_NODE（1）
Entity	实体名	null	ENTITY_NODE（6）
EntityReference	引用实体名	null	ENTITY_REFERENCE_NODE（5）
Notation	符号名	null	NOTATION_NODE（12）
ProcessionInstruction	目标	除目标以外的所有内容	PROCESSIONG_INSTRUCTION_NODE（7）
Text	#text	文本节点内容	TEXT_NODE（3）

3．获取文档中的指定元素

虽然通过遍历文档树中全部节点的方法，可以找到文档中指定的元素，但是这种方法比较麻烦，下面我们介绍两种直接搜索文档中指定元素的方法。

（1）通过元素的 ID 属性获取元素

document 对象的 getElementById() 方法可以通过元素的 ID 属性获取元素。例如，获取文档中 id 属性为 userId 的节点的代码如下：

```
document.getElementById("userId");
```

（2）通过元素的 name 属性获取元素

使用 document 对象的 getElementsByName() 方法可以通过元素的 name 属性获取元素。与 getElementsById() 方法不同的是，使用该方法的返回值为一个数组，而不是一个元素。如果想通过 name 属性获取页面中唯一的元素，可以通过获取返回数组中下标值为 0 的元素进行获取。例如，获取 name 属性为 userName 的节点的代码如下：

```
document.getElementsByName("userName")[0];
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2.4　上机实践

2.4.1　使用删除线样式标注商品特价

➤➤➤ 题目描述

本章介绍了 HTML 5 标记语言和 CSS 样式的内容。CSS 样式提供了一个 text-decoration 属性，用于给文本添加修饰效果，其中，有一种效果就是添加删除线样式。在显示商品特价信息时，通常将商品的原价采用删除线样式，表示该价格已经不使用了。本练习尝试编写一个 HTML 5 文档，要求创建 HTML 页面，实现使用删除线样式标注商品特价，效果如图 2.27 所示。

➤➤➤ 技术指导

在 CSS 样式中，提供了一个 text-decoration 属性，用于给文本添加修饰效果。如果将其属性值设置为 line-through，那么将为文本添加删除线样式，本上机实践的关键代码参考如下：

```
<span style="text-decoration:line-through"> 原价：796（元）</span>
```

2.4.2　将小写金额转换为大写金额

➤➤➤ 题目描述

在开发财务或与金融相关的网络管理系统时，经常会涉及金额的大小写转换问题。在输入时通常采用输入小写金额的方式以方便用户，但是数据可能存在被篡改的问题，解决方法是将用户输入的小写金额转换为大写，通过 JavaScript 就可以实现该功能。本练习要求，应用 JavaScript 实现一个将小写金额转换为大写金额的程序，运行结果如图 2.28 所示。输入小写金额，单击“转换”按钮，会转换为大写。



图 2.27　使用删除线样式标注商品特价



图 2.28　将小写金额转换为大写金额

➤➤➤ 技术指导

使用 JavaScript 实现将小写金额转换为大写金额的思路如下。

1）将大写的数字和单位分别保存在两个字符串中。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2）去除用户输入的小写金额中的小数点。

3）根据用户输入的小写金额长度取出该字符串所用到的单位。

4）取出小写字符对应的大写字符，并与单位组合成新的字符串；同时还要处理小写金额中连续多个 0 的情况。

2.4.3 验证用户注册信息的合法性

➤➤➤ 题目描述

本章介绍了 HTML 5 标记语言、CSS 样式和 JavaScript 脚本语言的内容。本练习尝试编写一个验证用户注册信息合法性的程序，要求应用 DIV+CSS 设计用户注册界面，并应用 JavaScript 验证输入数据的合法性，效果如图 2.29 所示。



图 2.29　验证用户注册信息的合法性

➤➤➤ 技术指导

实现验证用户注册信息的合法性，主要是应用 DIV+CSS 设计一个用户注册界面，然后编写自定义的 JavaScript 函数验证用户输入的注册信息。其中，需要验证的用户注册信息主要有检测表单元素是否空、验证用户名是否合法、验证密码是否合法，以及验证 E-mail 地址是否合法等。

本上机实践的关键代码如下。

```
//检测全部表单元素是否为空
function checkBlank(Form){
   var v=true;
   for(i=0;i<Form.length;i++){
          if(Form.elements[i].value == ""){      //Form 的属性 elements 的首字 e 要小写
                alert(Form.elements[i].title + "不能为空!");
                Form.elements[i].focus();        //指定表单元素获取焦点
                v=false;
                return false;
          }
   }
   return v;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
}
// 验证用户名是否合法
function checkusername(username){
   var str=username;
    // 在 JavaScript 中，正则表达式只能使用 "/" 开头和结束，不能使用双引号
   var Expression=/^(\w){3,20}$/;
   var objExp=new RegExp(Expression);                    // 创建正则表达式对象
   return objExp.test(str)                               // 通过正则表达式验证
}
// 验证密码是否合法
function checkPWD(PWD){
 var str=PWD;
    // 在 JavaScript 中，正则表达式只能使用 "/" 开头和结束，不能使用双引号
   var Expression=/^[A-Za-z]{1}([A-Za-z0-9]|[._]){5,19}$/;
   var objExp=new RegExp(Expression);                    // 创建正则表达式对象
   return objExp.test(str)                               // 通过正则表达式验证
 }
// 验证 E-mail 地址是否合法
function checkemail(email){
 var str=email;
    // 在 JavaScript 中，正则表达式只能使用 "/" 开头和结束，不能使用双引号
   var Expression=/^w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/;
   var objExp=new RegExp(Expression);                    // 创建正则表达式对象
   return objExp.test(str)                               // 通过正则表达式进行验证
}
```

2.5　小结

本章主要介绍了进行 Java Web 开发前，需要掌握的 HTML 语言、CSS 样式和 JavaScript 脚本，其中，HTML 语言主要用于描述网页文档的内容，它通过标记符号来标记要显示网页中的各个部分；CSS 样式主要是用于对页面内容进行美化，通过 HTML 和 CSS 的结合，可以设计出非常漂亮的网页界面；JavaScript 脚本语言主要用于客户端交互，例如，验证表单数据等。本章介绍的内容将对以后的 Java Web 开发打下良好的基础，希望读者认真学习，多多实践，力求更好地掌握本章的内容。

2.6　习题

选择题

1. 下面这些元素中，哪些是 HTML 5 新增的元素？（　　）

 A. <head>　　B. <header>

 C. <body>　　D. <embed>

2. 在表单中添加单选按钮时，需要设置 <input> 标记的 type 属性值为（　　）。

 A. radio　　B. radiobutton

 C. checkbox　　D. button

3. 在表单中添加隐藏域时，需要设置 <input> 标记的 type 属性值为（　　）。

 A. file　　B. text　　C. hidden

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

4．在 JavaScript 中，下面描述错误的是（　　）。

A．break 语句通常用在 for、while、do...while 或 switch 语句中。

B．continue 语句通常应用在 for、while、do...while 或 switch 语句中。

C．continue 语句不能应用在 switch 语句中。

D．以上都是正确的。

5．下面哪些是 JavaScript 支持的合法注释（　　）。

A．<!--　　-->　　B．//　　C．#　　D．/*　*/

填空题

1．无序列表是在每个列表项的前面添加一个_____。通过_____标记可以创建一组无序列表，其中每一个列表项以_____标记表示。

2．有序列表是在每个列表项目的前面加上_____。有序列表的标记为_____，每一个列表项前使用_____标记。

3．在 CSS 样式中，使用起始标记_____和结束标记_____来定义注释。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 3 章　驾驭 Java Web 开发环境

视频讲解：25 分钟

工欲善其事，必先利其器。在创建和开发一个动态网站之前，必须安装和配置 Web 应用程序的运行环境、服务器及相关的开发工具。

首先需要安装 JDK，其作用是提供 Java 程序的运行环境及 Java 开发工具包；其次需要安装 Web 服务器，本书选用的 Web 服务器是常用的 Tomcat 服务器。

通过学习本章，读者可以达到以下学习目的：

- 了解 Java Web 需要的开发环境；
- 学会安装与配置 JDK；
- 学会安装与配置 Tomcat 服务器；
- 掌握下载并安装 Eclipse 的方法；
- 熟悉并使用 Eclipse 开发工具。

3.1　Java Web所需要的开发环境

在搭建 Java Web 开发环境时，首先需要安装开发工具包 JDK（Java Develop Kit）、Web 服务器和数据库。为了提高开发效率，通常还需要安装 IDE（集成开发环境）工具。Java Web 开发环境如图 3.1 所示。

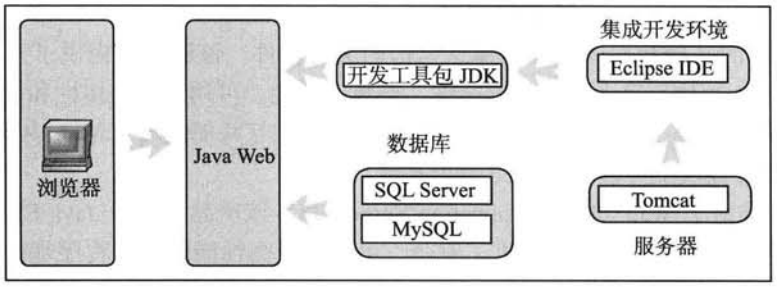


图 3.1　Java Web 应用的开发环境

3.1.1　开发工具包 JDK

JDK 包括运行 Java 程序所必需的 JRE（Java Runtime Environment，Java 的运行环境）及开发过程中常用的库文件，目前的最新版本为 jdk-7u3。

JDK、JRE 与 JVM 的区别如下。

JDK 中包括很多用 Java 编写的开发工具（如 javac.exe 和 jar.exe 等）和一个 JRE。如果计算机安装了 JDK，它会有两套 JRE，一套位于 jre 目录下，另一套位于 Java 目录下，后者少了服务器端的 Java 虚拟机。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

JRE 是面向 Java 程序的使用者，即 Java 程序则必须有 JRE 才能运行。

JVM 是 Java 虚拟机，在 JRE 的 bin 目录下有两个子目录（server 和 client），这是真正的 jvm.dll 所在。jvm.dll 无法单独工作，当 jvm.dll 启动后会使用 explicit 的方法，而这些辅助的动态链接库（.dll）都必须位于 jvm.dll 所在目录的父目录中。因此需要使用哪个 JVM，则在环境变量中设置 path 参数指向 JRE 所在目录下的 jvm.dll 即可。正是有了 JVM，Java 才实现了其跨平台性。JVM 屏蔽了与具体操作系统有关的信息，即在不同操作系统上虚拟出一个相同且只适合运行 Java 字节码文件的虚拟系统。

现在我们可以看出这样一个关系，JDK 包含 JRE，而 JRE 包含 JVM。

说明 *在需要开发 Java Web 应用的计算机中需要安装 JDK（包括 JRE），而在只需要发布并运行 Java Web 应用的计算机上，则可以只安装 JRE。*

3.1.2 Web 服务器

Web 服务器是运行及发布 Web 应用的容器，只有将开发的 Web 项目放置到其中，才能使网络中的所有用户通过浏览器访问。开发 Web 应用采用的服务器主要是 Servlet 兼容的 Web 服务器，比较常用的如下。

1. BEA WebLogic 服务器

WebLogic 是 BEA 公司的产品，分为 WebLogic Server、WebLogic Enterprise 和 WebLogic Portal 系列。其中 WebLogic Server 的功能特别强大，支持企业级、多层次和完全分布式的 Web 应用，配置简单且界面友好。对于那些正在寻求能够提供 Java 平台所拥有的一切应用服务器的用户来说，WebLogic 是一个十分理想的选择。

2. IBM WebSphere 应用服务器

IBM WebSphere 应用服务器即 IBM WebSphere Application Server（WAS），是 IBM WebSphere 软件平台的基础和面向服务的体系结构的关键构件。该服务器提供了一个丰富的应用程序部署环境，包括用于事务管理、安全性、群集、性能、可用性、连接性和可伸缩性等全套的应用程序服务。它与 Java EE 兼容，并为可与数据库交互并提供动态 Web 内容的 Java 组件、XML 和 Web 服务提供了可移植的 Web 部署平台。

目前 IBM 推出了 WebSphere Application Server V8，该产品是基于 Java EE 6 认证且支持 EJB 3.0 技术的应用程序平台。它提供了安全、可伸缩及高性能的应用程序基础架构，这些基础架构是实现 SOA 所需要的，提高了业务灵活性。

3. Tomcat 服务器

Tomcat 服务器最为流行，它是 Apache-Jarkarta 开源项目中的一个子项目，是一个小型、轻量级并支持 JSP 和 Servlet 技术的 Web 服务器，它已经成为学习开发 Java Web 应用的首选。本书将以其作为 Web 服务器，目前 Tomcat 的最新版本为 apache-tomcat-7.0.27。

3.1.3 数据库

开发动态网站时数据库是必不可少的，它主要用来保存网站中需要的信息。根据网站的规模应采用合适的数据库，如大型网站可采用 Oracle，中型网站可采用 Microsoft SQL Server

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

或MySQL，小型网站则可以采用Microsoft Access。Microsoft Access数据库的功能远不如Microsoft SQL Server和MySQL强大，但它具有方便和灵活的特点，对于一些小型网站来说是比较理想的选择。

3.1.4　Web浏览器

浏览器是客户端用户访问Web的应用工具，与开发JSP应用关系不大。所以开发JSP对浏览器的要求并不是很高，任何支持HTML的浏览器都可以。目前比较流行的Web浏览器是IE和火狐浏览器。

3.2　安装和配置JDK

在开发Java Web网站之前，需要先安装和配置JDK，下面将具体介绍下载并安装JDK和配置环境变量的方法。

3.2.1　下载JDK

由于推出JDK的Sun公司已经被Oracle公司收购了，所以JDK可以到Oracle官方网站（http://www.oracle.com/index.html）中下载。下面以目前最新版本的JDK 7 Update 3为例介绍下载JDK的方法，具体下载步骤如下。

Step 1）打开IE浏览器，在地址栏中输入URL地址“http://www.oracle.com/index.html”，并按下〈Enter〉键，进入到图3.2所示的Oracle官方网站页面。



图3.2　Oracle官方主页

Step 2）在Oracle主页中Downloads选项卡的Popular Downloads栏目中，单击Java for Developers超级链接，进入到如图3.3所示的Java SE相关资源下载页面。

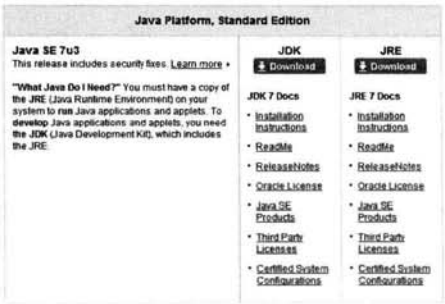
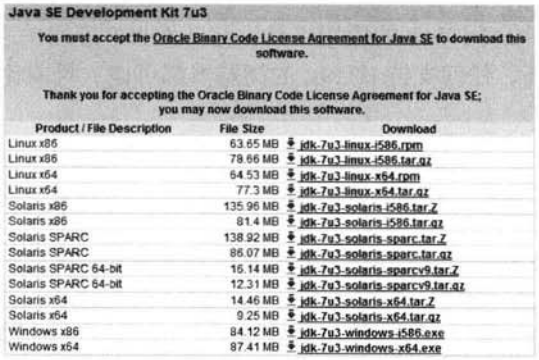


图3.3　JDK和JRE下载页面

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明 在 JDK 中，已经包含了 JRE。JDK 用于开发 Java 程序，JRE 用于运行 Java 程序。

Step 3) 在图 3.3 所示的页面中，单击 JDK 下方的 Download 按钮，将进入到如图 3.4 所示的下载列表页面，同意协议并选择适合当前系统版本的 JDK 下载。



Java SE Development Kit 7u3

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux x86	63.65 MB	jdk-7u3-linux-i586.rpm
Linux x86	78.66 MB	jdk-7u3-linux-i586.tar.gz
Linux x64	64.53 MB	jdk-7u3-linux-x64.rpm
Linux x64	77.3 MB	jdk-7u3-linux-x64.tar.gz
Solaris x86	135.96 MB	jdk-7u3-solaris-i586.tar.Z
Solaris x86	81.4 MB	jdk-7u3-solaris-i586.tar.gz
Solaris SPARC	138.92 MB	jdk-7u3-solaris-sparc.tar.Z
Solaris SPARC	86.07 MB	jdk-7u3-solaris-sparc.tar.gz
Solaris SPARC 64-bit	16.14 MB	jdk-7u3-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	12.31 MB	jdk-7u3-solaris-sparcv9.tar.gz
Solaris x64	14.46 MB	jdk-7u3-solaris-x64.tar.Z
Solaris x64	9.25 MB	jdk-7u3-solaris-x64.tar.gz
Windows x86	84.12 MB	jdk-7u3-windows-i586.exe
Windows x64	87.41 MB	jdk-7u3-windows-x64.exe

图 3.4 JDK 资源选择页面

3.2.2 安装 JDK

JDK 的安装文件下载后，就可以安装 JDK 了，具体的安装步骤如下。

Step 1) 双击刚刚下载的安装文件，将弹出如图 3.5 所示的欢迎对话框。

Step 2) 单击“下一步”按钮，将弹出“自定义安装”对话框，在该对话框中，可以选择安装的功能组件，这里选择默认设置，如图 3.6 所示。



图 3.5 欢迎对话框

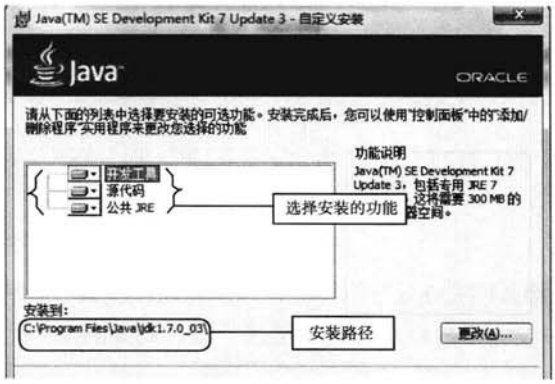


图 3.6 JDK“自定义安装”对话框

Step 3) 单击“更改”按钮，将弹出“更改文件夹”对话框，在该对话框中将 JDK 的安装路径更改为 C:\Java\jdk1.7.0_03\，如图 3.7 所示，单击“确定”按钮，将返回到自定义安装对话框中。

Step 4) 单击“下一步”按钮，开始安装 JDK。在安装过程中会弹出 JRE 的“目标文件夹”对话框，这里更改 JRE 的安装路径为 C:\Java\jre7\，然后单击“下一步”按钮，安装向导会继续完成安装进程。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明　JRE 全称为 Java Runtime Environment，它是 Java 运行环境，主要负责 Java 程序的运行，而 JDK 包含了 Java 程序开发所需要的编译、调试等工具，另外还包含了 JDK 的源代码。

Step 5　安装完成后，将弹出如图 3.8 所示的对话框，单击“继续”按钮，将安装 JavaFX SDK。如果不想安装，可以单击“关闭”按钮，取消 JavaFX 的安装。

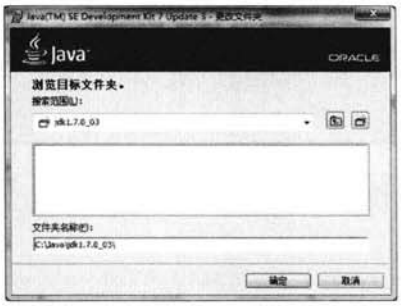


图 3.7　更改 JDK 的安装路径



图 3.8　“完成”对话框

说明　JavaFX 2.0 是 Oracle 公司发布的一款为企业业务应用提供的先进 Java 用户界面（UI）平台，它能帮助开发人员无缝地实现与本地 Java 功能及 Web 技术动态能力的混合与匹配。

3.2.3　在 Windows 系统下配置和测试 JDK

JDK 安装完成后，还需要在系统的环境变量中进行配置，下面将以在 Windows 7 系统中配置环境变量为例来介绍 JDK 的配置和测试，具体步骤如下。

Step 1　在“开始”菜单的“计算机”图标上单击鼠标右键，在弹出的快捷菜单中选择“属性”命令，在弹出的“属性”对话框左侧单击“高级系统设置”超链接，将出现如图 3.9 所示的“系统属性”对话框。

Step 2　单击“环境变量”按钮，将弹出“环境变量”对话框，如图 3.10 所示，单击“系统变量”栏中的“新建”按钮，创建新的系统变量。



图 3.9　“系统属性”对话框



图 3.10　“环境变量”对话框

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Step 3 弹出“新建系统变量”对话框，分别输入变量名“JAVA_HOME”和变量值（即 JDK 的安装路径），图中显示的变量值是笔者的 JDK 安装路径，读者需要根据自己的计算机环境进行修改，如图 3.11 所示。单击“确定”按钮，关闭“新建系统变量”对话框。

Step 4 在图 3.10 所示的“环境变量”对话框中双击 Path 变量对其进行修改，在原变量值最前端添加“.;%JAVA_ HOME%\bin;”变量值（注意：最后的“;”不要丢掉，它用于分隔不同的变量值），如图 3.12 所示。单击“确定”按钮完成环境变量的设置。

注意 不能删除系统变量 Path 中的原有变量值，并且“%JAVA_ HOME%\bin”与原有变量值之间用英文半角的“;”号分隔，否则会产生错误。

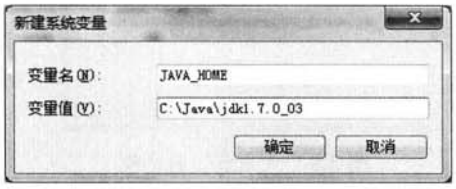


图 3.11 “新建系统变量”对话框



图 3.12 设置 Path 环境变量值

Step 5 JDK 安装成功之后必须确认环境配置是否正确。在 Windows 系统中测试 JDK 环境需要选择“开始”/“运行”命令（没有“运行”命令可以按〈Windows+R〉组合键），然后在“运行”对话框中输入 cmd 并单击“确定”按钮启动控制台。在控制台中输入 javac 命令，按〈Enter〉键，将输出如图 3.13 所示的 JDK 的编译器信息，其中包括修改命令的语法和参数选项等信息。这说明 JDK 环境搭建成功。

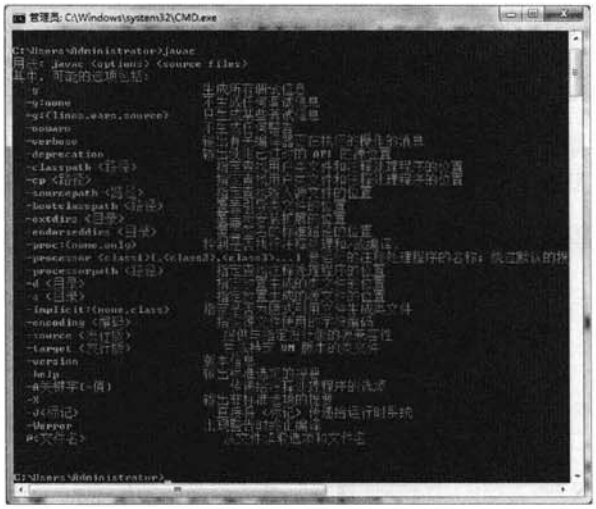


图 3.13 JDK 的编译器信息

3.3 安装和配置Tomcat

Tomcat 服务器是 ApacheJakarta 项目组开发的产品。当前的最新版本是 Tomcat 7，它能够

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

支持 Servlet 3.0 和 JSP 2.2 规范，并且具有免费和跨平台等诸多特性。Tomcat 服务器已经成为学习开发 Java Web 应用的首选，本书使用的就是 Tomcat 服务器。下面介绍 Tomcat 服务器的安装与配置。

3.3.1　获取 Tomcat

可以到它的官方网站（http://tomcat.apache.org）中下载 Tomcat 服务器，具体的下载步骤如下。

Step 1）在 IE 地址栏中输入 http://tomcat.apache.org，进入 Tomcat 官方网站，如图 3.14 所示，单击 Download 超链接。



图 3.14　Tomcat 官方网站首页

Step 2）进入 Tomcat 7 下载页面，如图 3.15 所示。

Step 3）显示 Tomcat 7.0.29 的下载列表，单击 Core 节点下的 zip(pgp,md5) 超链接，如图 3.16 所示。

说明　“zip”与“Windows Service Installer”对应的是 Windows 系统下的版本，其中“zip”下载后是一压缩文件，解压即可使用；“Windows Service Installer”是 Windows 下的安装版本，下载后需要安装。而 tar.gz 为 Linux 平台下的开发包。



图 3.15　Tomcat 7 下载页面

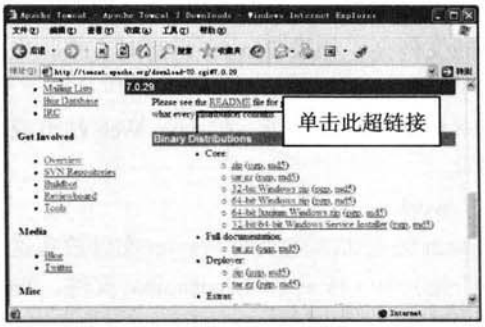


图 3.16　Tomcat 7.0.29 的下载列表

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Step 4 打开文件下载对话框，单击“保存”按钮，即可将 Tomcat 的安装文件下载到本地计算机中。

3.3.2 熟悉 Tomcat

下载 Tomcat 后，解压缩获取的压缩包，显示如图 3.17 所示的 Tomcat 目录与文件。

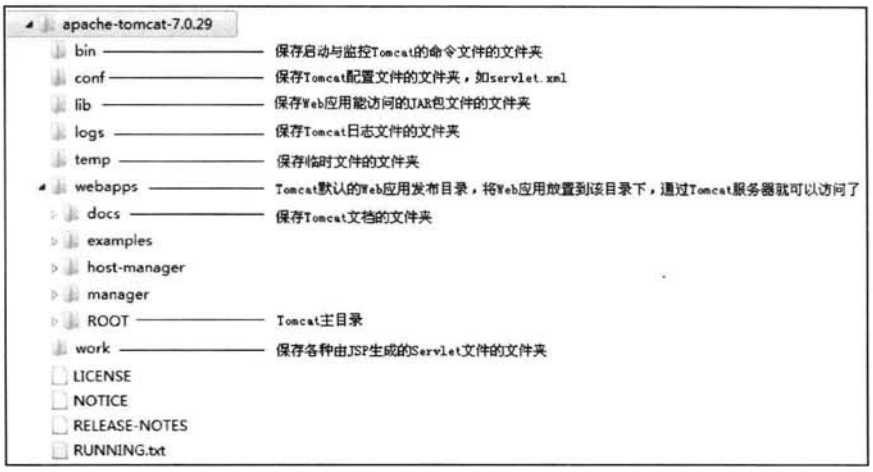


图 3.17 Tomcat 目录与文件

Tomcat 目录中的关键目录及文件的作用如下。

（1）bin

存放各种平台下启动和关闭 Tomcat 的脚本文件，其中包含两个关键文件 startup.bat、shutdown.bat，前者是 Windows 下启动 Tomcat 的文件；后者是 Windows 下关闭 Tomcat 的文件。

（2）conf

用于保存 Tomcat 的配置文件，其中 server.xml 为服务器的主配置文件；web.xml 为所有 Web 应用的配置文件；tomcat-users.xml 用于定义 Tomcat 配置用户的权限与安全。

（3）lib

存放 Tomcat 服务器和所有 web 应用都能访问的 JAR 文件。

（4）logs

存放 Tomcat 的日志文件，Tomcat 的所有日志文件都保存在其中。

（5）temp

临时文件夹，如果 Tomcat 运行时有临时文件，则保存在此目录中。

（6）webapps 目录

Web 应用的发布目录，把 Java Web 站点或 war 文件放入这个目录下，即可通过 Tomcat 服务器访问。

（7）work

Tomcat 解析 JSP 生成的 Servlet 文件放在这个目录中。

双击运行 bin 目录中的 startup.bat 文件，Tomcat 的运行窗口如图 3.18 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

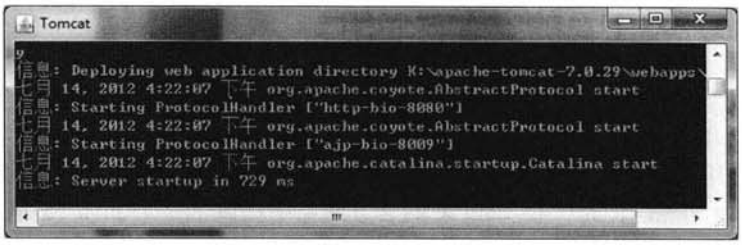


图 3.18　Tomcat 运行窗口

在 Tomcat 启动成功后，在浏览器的地址栏中输入“http://localhost:8080”访问 Tomcat。如果显示如图 3.19 所示的页面，则证明 Tomcat 运行成功。

说明　Tomcat 的默认端口为 8080，所以在访问 Tomcat 时需要在访问地址中设置此端口号。

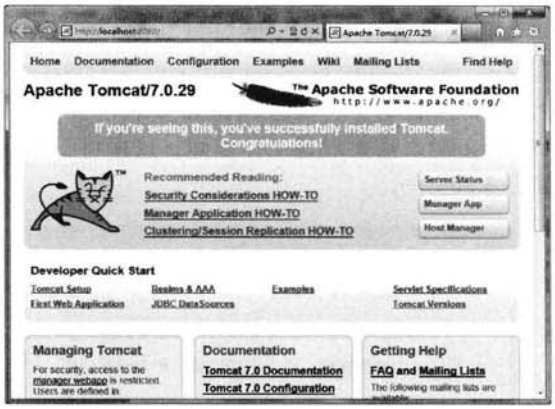


图 3.19　通过浏览器访问 Tomcat

3.3.3　修改 Tomcat 的默认端口

Tomcat 默认的服务端口为 8080，但该端口不是 Tomcat 唯一的端口，可以在安装过程中进行修改，如果在安装过程中没有进行修改，还可以通过 Tomcat 的配置文件进行修改。下面将介绍通过 Tomcat 的配置文件修改其默认端口的步骤。

Step 1）采用记事本打开 Tomcat 安装目录下的 conf 文件夹下的 servlet.xml 文件。

Step 2）在 servlet.xml 文件中找到以下代码：

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
```

Step 3）将上面代码中的 port="8080" 修改为 port="8081"，即可将 Tomcat 的默认端口设置为 8081。

说明　在修改端口时，应避免与公用端口冲突。建议采用默认的 8080 端口，不要修改，除非 8080 端口被其他程序所占用。

Step 4）修改成功后，为了使新设置的端口生效，还需要重新启动 Tomcat 服务器。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

3.4 Eclipse开发工具的安装与使用

Eclipse 是一个基于 Java 的、开放源码的、可扩展的应用开发平台，它为编程人员提供了一流的 Java 集成开发环境（Integrated Development Environment，IDE）。它是一个可以用于构建集成 Web 和应用程序开发工具的平台，其本身并不会提供大量的功能，而是通过插件来实现程序的快速开发功能。但是，在 Eclipse 的官方网站中提供了一个 Java EE 版的 Eclipse IDE。应用 Eclipse IDE for Java EE，可以在不需要安装其他插件的情况下创建动态 Web 项目。

3.4.1 Eclipse 的下载与安装

可以从官方网站下载最新版本的 Eclipse，具体网址为 http://www.eclipse.org。目前最新版本为 Eclipse 3.7.2，下载后的安装文件是 eclipse-jee-indigo-SR2-win32.zip。

Eclipse 的安装比较简单，只需要将下载到的压缩包解压缩到自己喜欢的文件夹中，即可完成 Eclipse 的安装。

3.4.2 启动 Eclipse

Eclipse 安装完成后，就可以启动了。双击 Eclipse 安装目录下的 eclipse.exe 文件，即可启动 Eclipse，在初次启动 Eclipse 时，需要设置工作空间，这里将工作空间设置在 Eclipse 根目录的 workspace 目录下，如图 3.20 所示。



图 3.20 设置工作空间

在每次启动 Eclipse 时，都会弹出设置工作空间的对话框，如果想在以后启动时，不再进行工作空间设置，可以勾选 Use this as the default and do not ask again 复选框。单击 OK 按钮后，即可启动 Eclipse，进入到如图 3.21 所示的界面。

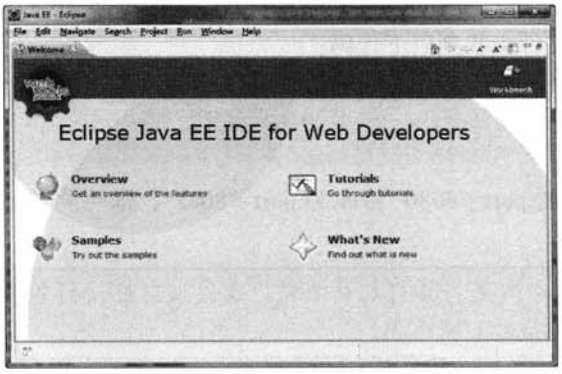


图 3.21 Eclipse 的欢迎界面

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

3.4.3　安装 Eclipse 中文语言包

直接解压完的 Eclipse 是英文版的，为了适应国际化的需求，Eclipse 提供了多国语言包，我们只需要下载对应语言环境的语言包，就可以实现 Eclipse 的本地化。例如，我们当前的语言环境为简体中文，就可以下载 Eclipse 提供的中文语言包。Eclipse 提供的多国语言包可以到 http://www.eclipse.org/babel/ 中下载。在该网站中，可以找到所用 Eclipse 版本对应的中文语言包，例如，Eclipse 3.7 所对应的中文语言包的下载页面如图 3.22 所示。

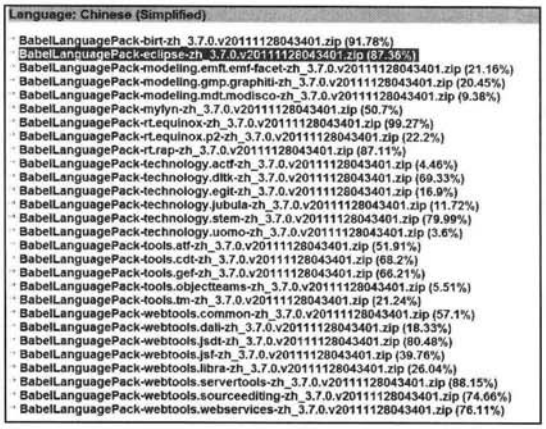


图 3.22　Eclipse 3.7 的中文语言包下载页面

单击图 3.22 所示的各个超链接，即可以下载对应的中文语言包。我们可以下载全部的中文语言包，也可以根据需要下载一部分。中文语言包下载后，将下载的所有语言包解压缩并覆盖 Eclipse 文件夹中同名的两个文件夹 features 和 plugins，这样在启动 Eclipse 时便会自动加载这些语言包。

3.4.4　Eclipse 工作台

启动 Eclipse 后，关闭欢迎界面，将进入到 Eclispe 的主界面，即 Eclipse 的工作台窗口。Eclipse 的工作台主要由菜单栏、工具栏、透视图工具栏、透视图、项目资源管理器视图、大纲视图、编辑器和其他视图组成。Eclipse 的工作台如图 3.23 所示。

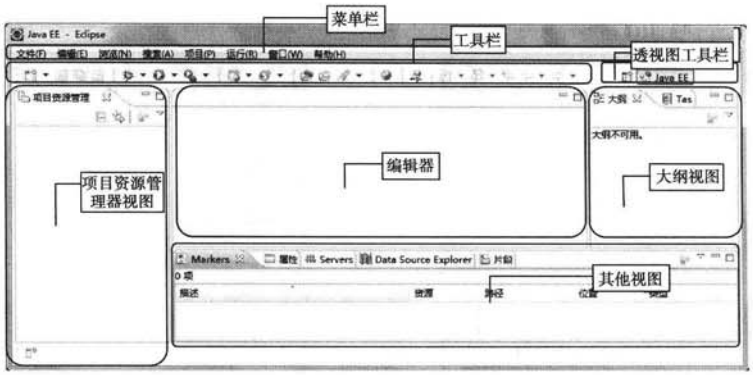


图 3.23　Eclipse 的工作台

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明 在应用 Eclipse 时，各视图的内容会有所改变，例如，打开一个 JSP 文件后，在大纲视图中，将显示该 JSP 文件的节点树。

3.5 做好项目开发的准备工作

在获取 Tomcat 服务器和 Eclipse IDE 开发工具后，需要通过配置 Eclipse 与 Tomcat 来开发 Java Web 项目。

3.5.1 集成 Eclipse 与 Tomcat

为了提高开发效率，需要将 Tomcat 服务器配置到 Eclipse 之中，为 Web 项目指定一台 Web 应用服务器。然后即可在 Eclipse 中操作 Tomcat 并自动部署和运行 Web 项目，操作步骤如下。

Step 1 在 Eclipse 中，单击“窗口”/“首选项”选项，打开 Eclipse 的“首选项”窗口，如图 3.24 所示。

Step 2 选择“服务器”/“运行时环境”选项，如图 3.25 所示。

Step 3 单击“添加”按钮，添加 Tomcat 服务器，弹出“新建服务器运行时”窗口，如图 3.26 所示。

说明 在“新建服务器运行时”窗口中有多种类型的 Web 应用服务可供选择，此处可根据实际应用选择。

Step 4 选择 Apache/Apache Tomcat v7.0 选项，单击“下一步”按钮，打开“指定安装目录”窗口，如图 3.27 所示。



图 3.24 “首选项”窗口

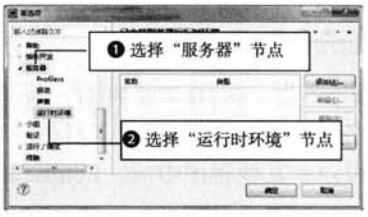


图 3.25 选择“服务器”/“运行时环境”选项



图 3.26 “新建服务器运行时”窗口



图 3.27 “指定安装目录”窗口

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明　由于前面下载的是 Tomcat 7.0.29 版本，所以此处选择 Apache Tomcat v7.0 选项，与下载的版本相对应。

Step 5）单击“浏览”按钮，显示“浏览文件夹”对话框，如图 3.28 所示。

说明　指定的 Tomcat 目录必须是 Tomcat 的根目录，实例中将 Tomcat 放置在 K 盘根目录下。

Step 6）选择 Tomcat 目录后，单击“指定安装目录”窗口中的“完成”按钮。



图 3.28　“浏览文件夹”对话框

3.5.2　完善 Web 项目开发所需配置

在集成 Eclipse 与 Tomcat 后，还需要完成如下两项配置。

1．为 Eclipse 指定 Web 浏览器

在默认情况下，Eclipse 在工作台中使用系统默认的 Web 浏览器浏览网页。但在开发过程中，这样有些不方便，通常我们会为其指定一种浏览器，并且在 Eclipse 的外部打开，具体的操作步骤如下。

Step 1）选择 Eclipse 界面窗中的“窗口”/“首选项”选项，依次选择“常规”/“Web 浏览器”选项，如图 3.29 所示。

Step 2）选择 Use external web browser 单选按钮和 Internet Explorer 复选框，然后单击“确定”按钮。



图 3.29　选择“常规”/“Web 浏览器”选项

说明　实例中选择的是 IE 浏览器，也可选择计算机中已经安装的其他浏览器。

2．指定 JSP 页面的编码格式

默认情况下，在 Eclipse 中创建的 JSP 页面是“ISO-8859-1”编码格式。此格式不支持中文字符集，所以需要为其指定一个支持中文的字符集。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

1）选择 Eclipse 界面中的“窗口”/“首选项”选项，依次选择 Web/“JSP 文件”选项，如图 3.30 所示。



图 3.30　选择 Web/“JSP 文件”选项

2）在“将使用下列编码”下拉列表框中选择“ISO 10646/Unicode(UTF-8)”选项，单击“确定”按钮。

说明　UTF-8 编码是一个支持中文的字符集合。

3.6　使用Eclipse开发一个JSP网站

Eclipse 安装完成后，就可以在 Eclipse 中开发 Web 应用了。在 Eclipse 中开发 JSP 网站的基本步骤如下。

【例 3.1】　创建一个 JSP 网站，在页面中输出文字“坚韧、创新、博学、笃行”，具体操作步骤如下：

实例位置：光盘 \MR\ 源码 \ 第 3 章 \3.1

Step 1）启动 Eclipse，并选择一个工作空间，进入到 Eclipse 的工作台界面。

Step 2）在工具栏上选择“文件”/“新建”/Dynamic Web Project 菜单项，将打开“新建动态 Web 项目”对话框，在该对话框的 Project name 文本框中输入项目名称，这里为 firstProject，在 Dynamic Web module version 下拉列表中选择 3.0；在“配置”下列表框中选择已经配置好的 Tomcat 服务器（这里为 Tomcat 7.0），其他采用默认，如图 3.31 所示。

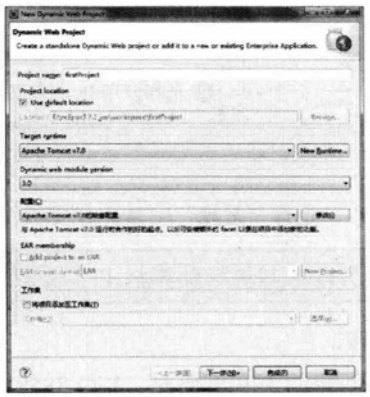


图 3.31　“新建动态 Web 项目”对话框

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Step 3) 单击“下一步”按钮，将打开配置 Java 应用的对话框（这里采用默认），再单击“下一步”按钮，将打开如图 3.32 所示的“配置 Web 模块设置”对话框，这里采用默认。

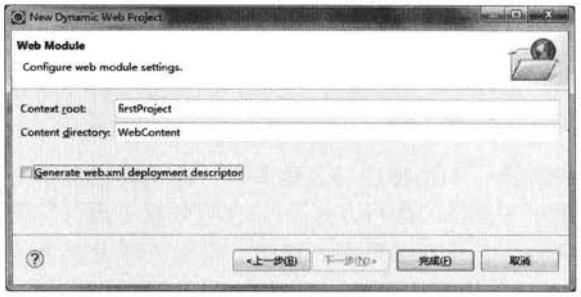


图 3.32　“配置 Web 模块设置”对话框

说明　在图 3.32 中，如果采用默认设置，新创建的项目将不自动创建 web.xml 文件，如果需要自动创建该文件，那么可以选中 Generate web.xml deployment descriptor 复选框。

Step 4) 单击“完成”按钮，完成项目 firstProject 的创建。这时，在 Eclipse 的“项目资源管理器”中，将显示新创建的项目。

Step 5) 在 Eclipse 的“项目资源管理器”中，选中 firstProject 节点下的 WebContent 节点，并单击鼠标右键，在打开的快捷菜单中，选择“新建”/ JSP File 菜单项，打开 New JSP File 对话框，在该对话框的“文件名”文本框中输入文件名 index.jsp，其他采用默认设置，如图 3.33 所示。



图 3.33　New JSP File 对话框

Step 6) 单击“完成”按钮，完成 JSP 文件的创建。此时，在项目资源管理器的 WebContent 节点下，将自动添加一个名称为 index.jsp 的节点，同时，Eclipse 会自动以默认的与 JSP 文件关联的编辑器将文件在右侧的编辑窗口中打开。

Step 7) 将 index.jsp 文件中的默认代码修改为以下代码，并保存该文件。

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE HTML>
<html>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<head>
<meta charset="utf-8">
<title>使用 Eclipse 开发一个 JSP 网站 </title>
</head>
<body>
坚韧、创新、博学、笃行
</body>
</html>
```

Step 8) 在“项目资源管理器”中选择项目名称节点，在工具栏上单击 ▶ 按钮中的黑三角，在弹出的快捷菜单中选择“运行方式”/“在服务器上运行”菜单项，将打开“在服务器上运行”对话框，在该对话框中，选中“将服务器设置为项目缺省值（请不要再询问）”复选框，其他采用默认设置，如图 3.34 所示。



图 3.34 “在服务器上运行”对话框

Step 9) 单击“完成”按钮，即可通过 Tomcat 运行该项目。运行结果如图 3.35 所示。



图 3.35 运行结果

3.7 小结

环境搭建是进行 Java Web 开发的前奏。在本章中，首先介绍了 Java Web 应用所需的开发环境，之后详细地为大家介绍了 Tomcat 服务器的安装与配置、Eclipse 的下载与使用等内容，并通过一个具体的实例介绍使用 Eclipse 开发 Web 应用的具体过程。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

3.8　习题

选择题

1．下面哪个是 Tomcat 服务器的 Web 发布目录？（　　）

A．webapps　　B．work　　C．.bin　　D．lib

2．下面的服务器哪个不是 Java Web 服务器？（　　）

A．BEA WebLogic 服务器　　B．IIS 服务器

C．IBM WebSphere 应用服务器　　D．Tomcat 服务器

3．下面选项中哪一个是有效的字符编码格式？（　　）

A．GBK　　B．GBM　　C．GUI

4．Tomcat 服务器的默认端口为？（　　）

A．80　　B．90　　C．8080　　C．以上都不是

填空题

1．JDK 的全称是_____。

2．JDK、JRE 和 JVM 三者之间的关系是_____。

3．手动启动 Tomcat 服务器，运行的是_____文件。

4．在配置 JDK 的环境变量时，假如已经配置了 JAVA_HOME 变量，那么 PATH 变量的值应该是_____。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 4 章　Java Web 开发必修课之 JSP 语法

视频讲解：55 分钟

学习任何一种语言，首先要从其基础语法开始。熟悉 Java 的读者会很容易掌握 JSP（Java Server Pages）中的语法，因为一个 JSP 页面基本上是通过特殊的约定将 Java 代码嵌入到 HTML 标记中构成的，如脚本程序（Scriptlet）就是 JSP 中的一种语法规则。在脚本程序中包含的是任意的 Java 代码片段，所以 Java 中的一些语法规则会被应用到 JSP 中。当然这仅仅是 JSP 语法中的一小部分。本章将介绍 JSP 的基本语法。

通过学习本章，读者可以达到以下学习目的：

- 了解 JSP 页面的基本构成；
- 掌握 JSP 的指令标识；
- 掌握如何使用 JSP 脚本；
- 合理使用 JSP 的注释；
- 掌握常用的 JSP 动作标识。

4.1　了解JSP页面的基本构成

JSP 是基于 Java 语言的动态网页技术，它以“.jsp”为扩展名。在一个 JSP 页面中可以包括指令标识、HTML 代码、JavaScript 代码、嵌入的 Java 代码、注释和 JSP 动作标识等内容，但这些内容并不是一个 JSP 页面所必需的。下面通过一个简单的 JSP 页面说明 JSP 页面的构成。

【例 4.1】 创建一个名为 index.jsp 的页面，在其中输出当前的系统时间。index.jsp 文件的代码如下：

实例位置：光盘 \MR\ 源码 \ 第 4 章 \4.1

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ page import="java.util.Date"%>
<%@ page import="java.text.SimpleDateFormat"%>
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title> 一个简单的 JSP 页面——显示系统时间 </title>
</head>
<body>
<%
Date date = new Date();                          //获取日期对象
//设置日期时间格式
SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
String today = df.format(date);  //获取当前系统日期
%>
当前时间：<%=today%>                                          <!-- 输出系统时间 -->
```

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
</body>
</html>
```

<body>标签中的“<%”与“%>”包围部分为 Java 代码，其功能是创建一个 Date 对象并通过 SimpleDateFormat 对象格式化当前的时间。实例的运行结果如图 4.1 所示。

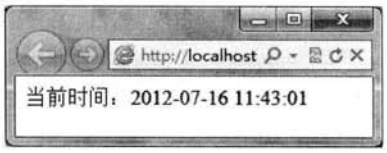


图 4.1　实例运行结果

可以看出在 JSP 页面中可以包含指令标识、HTML 代码、嵌入的 Java 代码和注释等内容，如图 4.2 所示。

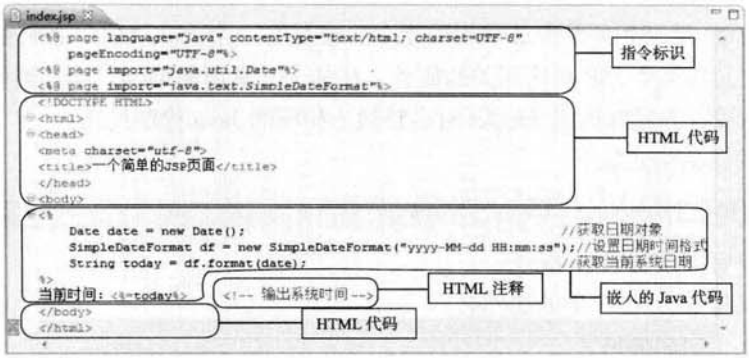


图 4.2　JSP 页面包含的内容

当 JSP 文件部署到 Web 应用服务器中后并不能直接回应给浏览器，其运行需要经历 4 个关键阶段，如图 4.3 所示。

当浏览器向 Web 应用服务器请求一个 JSP 页面时，Web 应用服务器将其转换为一个 Servlet 文件（即一个 .java 文件），然后将这个 Java 文件编译为一个字节码文件（即一个 .class 文件）。最后 Web 应用服务器加载转换后的 Servlet 实例，处理客户端的请求，并返回 HTML 格式的响应回应给客户端浏览器。

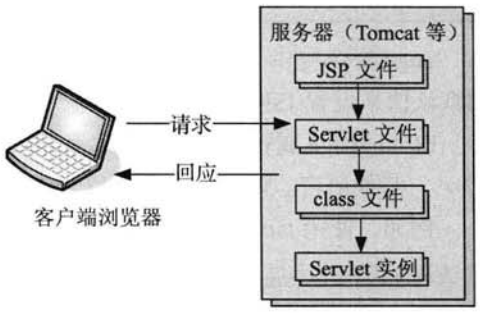


图 4.3　JSP 运行的 4 个关键阶段

说明　Servlet 定义在 Java EE API 中，它实质是一个用 Java 语言编写的类，但能够接收及回应 Web 请求。

4.2　使用JSP指令标识

指令标识主要用于设置整个 JSP 页面范围内有效的相关信息，这些标识会被服务器解释执行，但不会产生任何内容输出到网页中。

JSP 中包含了 page、include 和 taglib 共 3 个指令标识，其语法结构相同，定义的方法如下：

```
<%@ 指令名 属性 1="属性值 1" 属性 2="属性值 2" …%>
```

❑ 指令名：指定指令名，取值为 page、include 和 taglib 指令。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 属性：指定属性名称，不同的指令包含不同的属性。如果在一个指令中需要设置多个属性，则属性之间用逗号或空格分隔。
- 属性值：指定属性值。

说明 虽然不同的指令标识作用不同，但其语法结构相同。

4.2.1 使用页面指令 page

页面指令 page 是 JSP 页面中最常用的指令，也是每一个 JSP 页面中必须要用到的一个指令。其作用是定义与整个 JSP 页面相关的属性，比如 JSP 页面的编码、内容类型及引用的类库等，当这些属性被服务器解析成 Servlet 时会转换为相应的 Java 程序代码。

page 指令的语法格式如下：

```
<%@ page 属性 1="属性值 1" 属性 2="属性值 2" … %>
```

- page：声明指令为 page 指令。
- 属性：指定 page 指令中的属性名。
- 属性值：指定属性值。

注意 在使用一个指令时注意指令名与 @ 符号之间有一个空格。

page 指令提供了如下属性，在实际编程过程中，其中的很多属性可以省略，该指令会使用默认值来设置 JSP 页面。

- language 属性：指定当前页面中使用的语言，目前只支持 Java 语言，以后可能会支持其他语言。该属性的默认值为 Java。

例如，使用 language 属性设置 JSP 页面中使用的语言为 Java，代码如下：

```
<%@ page  language="java" %>
```

- contentType 属性：设置 JSP 页面的 MIME 类型和字符编码，浏览器会根据该属性指定的类型和编码显示网页内容。

例如，将页面的 MIME 类型设置为 HTML 文本且字符的编码设置为 UTF-8 的代码如下：

```
<%@ page  language="java" contentType="text/html; charset=UTF-8" %>
```

注意 在 Eclipse 中，JSP 页面默认的编码为不支持中文的 ISO-8859-1，如果未将其设置为中文编码（GBK、GB2312 或 UTF-8），运行时中文字符将出现乱码。

- pageEncoding 属性：设置 JSP 页面的编码格式，在 JSP 页面中所有代码都使用该属性指定的字符集。如果设置为 ISO-8859-1，则该 JSP 页面不支持中文字符。通常情况下，需要将其设置成为支持中文的字符集。例如，指定页面的编码格式为 UTF-8 的代码如下：

```
<%@ page  pageEncoding="UTF-8" %>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明 通常情况下，pageEncoding 和 contentType 属性会同时存在，前者的作用是指定 JSP 页面的编码格式，如果值设置为非支持中文的字符集（如 ISO-8859-1），那么在这个页面中不能编写中文信息；否则会出现乱码。后者的作用是指定服务器处理请求完成后，回应给浏览器的内容类型。如值设置为“text/html; charset=UTF-8”，则说明回应的页面是 HTML 文档或文本内容，其字符集为 UTF-8。

- import 属性：导入 JSP 页面中的类包，导入后在 JSP 页面中可以通过嵌入 Java 代码的方法使用这些类包。

例如，通过 import 属性导入 java.util.Date 类的代码如下：

```
<%@ page import="java.util.Date" %>
```

- buffer 属性：设置 out 对象使用的缓冲区大小，默认为 8 KB，其单位只能使用 KB。建议开发人员使用 8 的倍数（如 16、32、64 和 128 等）作为该属性的属性值。

例如，设置缓冲区的大小为 16 KB 的代码如下：

```
<%@ page  buffer="16KB" %>
```

- autoFlush 属性：指定当缓冲区已满时自动将缓冲区中的内容输出到客户端，默认值为 true。如果设置为 false，当缓冲区已满时将抛出 JSP Buffer overflow 异常。

例如，设置缓冲区已满时未自动将其内容输出到客户端的代码如下：

```
<%@ page  autoFlush="false" %>
```

注意 如果将 buffer 属性的值设置为 none，则 autoFlush 属性不能设置为 false。

- isErrorPage 属性：将当前 JSP 页面设置为错误处理页面，以处理另一个 JSP 页面的错误，即异常处理。只有在错误处理页面中将 isErrorPage 属性设置为 true 时，才可以调用 exception 对象输出错误信息。

例如，将当前页面设置为错误处理页面的代码如下：

```
<%@ page isErrorPage="true" %>
```

- errorPage 属性：指定当前页面出现异常时调用的另一个页面（即错误处理页面），在错误处理页面中必须将 isErrorPage 属性值设置为 true。

例如，指定当前页面出现异常时调用 error.jsp 页面显示错误信息的代码如下：

```
<%@ page errorPage="error.jsp" %>
```

在 error.jsp 页面需要通过以下面的代码将该页面设置为错误处理页面：

```
<%@ page isErrorPage="true" %>
```

然后在 error.jsp 页面中即可通过以下代码输出错误提示信息：

```
<%=exception.getMessage()%>
```

- session 属性：指定当前 JSP 页面是否支持 session，默认值为 true，即支持 session。
- isELIgnored 属性：指定是否禁用 EL 表达式，如果为 true，该页面将忽略 EL 表达式；否则将执行 EL 表达式。
- isThreadSafe 属性：指定 JSP 页面是否是线程安全的，如果为 true，则表明 JSP 页面在

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

同一时间可以被多个线程访问；否则不可。

4.2.2 使用文件包含指令 include

include 指令是 JSP 提供的页面包含指令，它可以将一个 JSP 页面包含到另一个 JSP 页面中实现 JSP 页面的重用，包含文件的过程如图 4.4 所示。

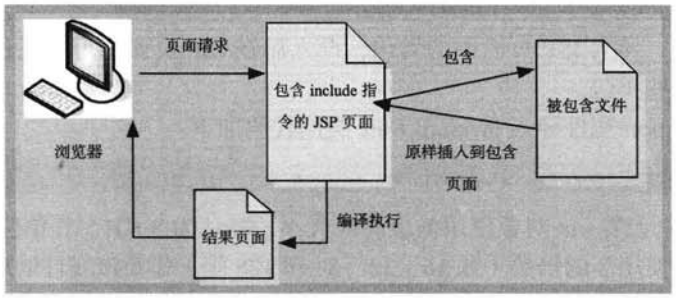


图 4.4 include 指令包含文件的过程

include 指令是静态包含，即被包含文件中所有内容会被原样包含到该 JSP 页面中。即使被包含文件中有 JSP 代码，在包含时也不会被编译执行。将两个页面组合成一个页面后编译处理，最后返回结果页面。

注意 使用 include 指令最终将生成一个文件，所以在被包含和包含的文件中不能有相同名称的变量；否则会发生编译错误。

include 指令的语法格式如下：

```
<%@ include file="path"%>
```

该指令只有一个 file 属性，用于指定要包含文件的路径，该路径可以是相对或绝对路径。

技巧 通常情况下，一个网站之中的页面头部（banner 部分）与底部（版权信息部分）都是相同的，可以分别创建这两个页面。然后通过 include 指令将其包含在其他页面中，从而减少冗余代码。

【例 4.2】 应用 include 指令将页面头部（top.jsp 文件）和版权信息（copyright.jsp 文件）包含到当前页面中。

实例位置：光盘 \MR\ 源码 \ 第 4 章 \4.2

Step 1) 编写一个名为 top.jsp 的文件，用于放置网站的 Banner 信息和导航栏，这里将 Banner 信息和导航栏设计为一张图片。要完成 top.jsp 文件，只需要在该页面中通过 标记引入图片即可。该文件的代码如下：

```
<%@ page pageEncoding="UTF-8"%>
<img src="images/banner.jpg">
```

Step 2) 编写一个名为 copyright.jsp 文件，用于放置网站的版权信息，该文件的代码如下：

```
<%@ page pageEncoding="UTF-8"%>
<div style="width:780px;height:102px;background-image: url(images/copyright.
jpg)"></div>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Step 3 创建一个名为 index.jsp 的文件，在该页面中包括 top.jsp 和 copyright.jsp 文件，从而实现一个完整的页面。index.jsp 文件的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"pageEncoding= "UTF
-8"%>
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>使用文件包含 include 指令 </title>
</head>
<body style="margin:0px;">
<div align="center">
<%@ include file="top.jsp"%>
<div style="width:780px;height:205px;background-image: url(images/center.jpg)">
</div>
<%@ include file="copyright.jsp"%>
</div>
</body>
</html>
```

运行程序，运行结果如图 4.5 所示。

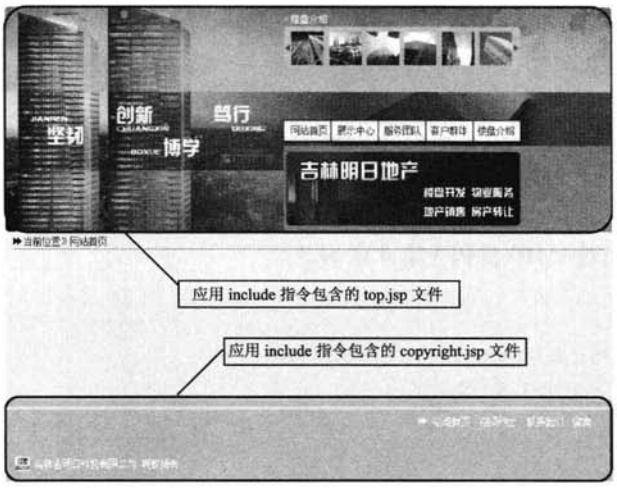


图 4.5　运行结果

4.2.3　使用引用标签库指令 taglib

taglib 指令用于声明一个标签的引用，在 JSP 页面之中声明了哪个标签库的引用，即可在 JSP 页面中调用哪个标签。该指令的语法格式如下：

```
<%@ taglib prefix="tagPrefix" uri="tagURI" %>
```

- taglib 属性：声明指令为 taglib 指令。
- prefix 属性：指定标签库的前缀。
- uri 属性：指定标签库文件的位置。

例如，引用 JSTL 中的核心标签库可以使用下面的代码：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

说明 在 JSP 页面中使用标签库可以简化 JSP 页面并减少 Java 代码，目前使用十分广泛，如常用的标签库有 JSTL、Struts 标签和 Spring 标签等。

4.3 使用JSP脚本

JSP 之所以应用灵活，其主要原因是在 JSP 页面中可以很方便地调用 Java 或脚本代码来实现所需功能。JSP 中的脚本标识包括 JSP 表达式（Expression）、声明标识（Declaration）和脚本程序（Scriptlet）。通过这些标识，在 JSP 页面中可以如编写 Java 程序一样来声明变量、定义方法或执行各种表达式的运算。

4.3.1 在 JSP 中应用代码片段

代码片段指在 JSP 页面中嵌入的 Java 或脚本代码，它将在页面请求的处理期间被执行。通过 Java 代码可以定义变量或是流程控制语句等，通过脚本代码可以应用 JSP 的内置对象在页面输出内容、处理请求和响应并访问 session 会话等。

代码片段的语法格式如下：

```
<% Java 代码或是脚本代码 %>
```

代码片段的使用比较灵活，它可以是 Java 代码，也可以调用 JSP 提供的内置对象。

【例 4.3】 创建一个名为 index.jsp 的页面，在其中应用代码片段输出 10 的阶乘。index.jsp 文件的关键代码如下：

实例位置：光盘 \MR\ 源码 \ 第 4 章 \4.3

```
<%
   int num = 1;                                    // 定义保存计算结果的变量
   // 通过 for 循环计算 10 的阶乘
   for(int i=1; i<=10; i++){
          num *= i;
   }
   out.print("10 的阶乘是: " + num);               // 输出 10 的阶乘
%>
```

在 index.jsp 页面中主要应用 for 循环计算 10 的阶乘，其中变量 num 用于保存每次计算的结果，for 循环结束后调用 JSP 内置对象 out 将计算结果 num 的值输出到页面中。

说明 JSP 内置对象 out 的 print() 方法将指定的对象输出到页面中。

实例运行后的结果如图 4.6 所示。

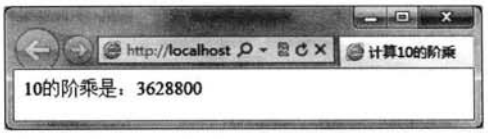


图 4.6 实例运行结果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

4.3.2　使用 JSP 表达式

JSP 表达式用于在页面中输出信息，它可以插入到网页的文本中用于输出文本内容，也可以插入到 HTML 标记中用于动态设置属性值。

JSP 表达式的语法格式如下：

```
<%= 表达式 %>
```

表达式可以是任何 Java 语言的完整表达式，其最终运算结果将被转换为字符串。

例如，使用 JSP 表达式输出用户相关信息的代码如下：

```
<%
    String name = "admin";                    // 用户名
    String sex = "男";                        // 性别
    String photo = "1.gif";                   // 照片
%>
用户名：<%=name %>
性别：<%=sex %>
照片 <img alt="照片" src="<%=photo %>">
```

上述代码首先使用 JSP 代码片段，分别将用户名赋值给 name 变量、性别赋值给 sex 变量并将照片赋值给 photo 变量。之后通过 JSP 表达式依次输出已声明的用户信息，运行上述的代码后将会形成如下 HTML 代码：

```
用户名：admin
性别：男
照片 <img alt="照片" src="1.gif">
```

注意　在 <% 与 = 之间不可以有空格；否则会出现编译错误，但 = 号与其后面的表达式之间可以有空格。

4.3.3　使用声明标识

在一个 JSP 页面中也可以如编写 Java 文件一样定义成员变量及成员方法，此种方式即声明标识的应用。使用声明标识在 JSP 页面中定义变量或方法是全局的，同时要求遵循 Java 语言的语法。

声明标识的语法格式如下：

```
<%! 声明变量或方法的代码 %>
```

说明　由于使用声明标识声明的变量是全局的，所以在多个用户并发访问时会产生线程安全的问题，此种方式应用很少。

【例 4.4】　应用 JSP 声明统计人数。index.jsp 文件的具体代码如下：

实例位置：光盘 \MR\ 源码 \ 第 4 章 \4.4

```
<%@ page language="java" contentType="text/html; charset=UTF-8"pageEncoding=
"UTF-8"%>
<!DOCTYPE html>
<html>
<head>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<meta charset="UTF-8">
<title>来访统计</title>
</head>
<body>
    <%!
            int count = 0;
            public int getCount(){
                    count ++;
                    return count;
            }
    %>
    您好，您是本站的第
    <%=getCount() %>
      位来访者！
</body>
</html>
```

在 index.jsp 页面中通过声明标识定义一成员变量 count 用于保存来访的人数，并调用 getCount() 方法获取当前来访人数值，最后通过 JSP 表达式来调用 getCount() 方法将来访人数的值输出到页面之中。

注意 由于使用声明标识并不能确保线程的安全，所以实例中通过 page 指令将 isThreadSafe 的值设置为 false。

实例运行后在页面中输出来访人数，通过两个浏览器打开 index.jsp 页面，运行结果如图 4.7 所示。



图 4.7 实例运行结果

4.4 巧用注释

在 JSP 页面中代码注释可分为基本 HTML 注释、隐藏注释和代码片段中的注释等，不同的注释适用于不同的位置，其作用也不同。

4.4.1 基本 HTML 注释

HTML 语言是 Web 开发中的基础，HTML 注释就是应用在 HTML 代码中的解释或说明性的文字，这些注释的内容并不会被浏览器解析执行。由于 JSP 页面中包含了 HTML 代码，所以在 JSP 页面同样可以使用 HTML 注释。

HTML 注释的语法格式如下：

```
<!-- 注释内容 -->
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在HTML注释中可以嵌入由JSP表达式指定的动态代码，即HTML注释还可以使用以下格式：

```
<!-- 注释内容 <%= 表达式 %> -->
```

例如，在一个JSP页面中添加以下注释代码：

```
<!-- 当前时间：<%=new java.util.Date().toLocaleString()%> -->
```

在JSP文件运行后通过查看HTML源代码可以得到类似的注释信息：

```
<!-- 当前时间：2012-07-19 14:58:31 -->
```

注意　HTML注释并不是安全的注释方式，可以通过查看网页源代码的方式查看编写的注释内容。

4.4.2　隐藏注释

JSP提供了一种隐藏注释，JSP容器会在转换编译JSP的过程中过滤注释的内容。所以在形成的HTML代码中不会出现注释内容，此种注释方式的安全性比较高。

隐藏注释的语法格式如下：

```
<%-- 注释内容 --%>
```

说明　开发涉及安全方面的信息并需要编写注释时应使用隐藏注释。

例如，在一个含有隐藏域的表单中不需要他人得知某个字段的含义，而又需要编写注释增强程序的可读性，即可使用隐藏注释：

```
<%-- 隐藏的用户 id（由系统分配） --%>
<input type="hidden" name="userId" value="1" />
<!-- 用户名 -->
<input type="text" name="userName" value="admin"/>
```

上述代码在Web应用服务器解析后使用隐藏注释的代码将被忽略，生成的HTML代码如下：

```
<input type="hidden" name="userId" value="1" />
<!-- 用户名 -->
<input type="text" name="userName" value="admin"/>
```

4.4.3　代码片段中的注释

在代码片段中编写注释与在Java代码中编写注释的方法相同，可分为单行与多行注释。其中单行注释以“//”开头，后面接注释内容，其语法格式如下：

```
// 注释内容
```

多行注释以“/*”开头，以“*/”结束。在这个标识中间的内容为注释内容，并且注释内容可以换行。其语法格式如下：

```
/*
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    注释内容 1
    注释内容 2
    ...
 */
```

此外在 JSP 中也可以使用 Java 代码中的规范形式注释，即 Java doc 的注释方法，以“/**”开头并以“*/”结束，其语法格式如下：

```
/**
  * 提示信息 1
  * 提示信息 2
  * ...
*/
```

说明 为了使编写程序的注释变得美观，通常情况下会在每行注释的前面加上一个“*”号。

使用此种注释方式的好处是当使用某一方法时，集成开发工具会给出相应的提示，如下面这段代码：

```
<%!
    /**
     * 获取当前时间
     * @return 当前时间格式化字符串
     */
    public String getCurrentTime(){
            // 实例化 SimpleDateFormat
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
            // 实例化 Date
            Date date = new Date();
            return sdf.format(date);
    }
%>
```

在 Eclipse 中当鼠标指针移动到 getCurrentTime() 方法上时会提示如图 4.8 所示的提示信息。



图 4.8 提示信息

4.5 常用JSP动作标识

在 JSP 规范中定义了一些标准的动作，用于为请求处理阶段提供信息。如操作 JavaBean、包含其他文件和执行请求转发等，这些动作在请求处理阶段按照在页面中出现的顺序执行。由于 JSP 动作标识基于 XML 语法实现，所以在 JSP 页面中只需要遵循 XML 语法调用即可。

JSP 动作标识的通用语法格式如下：

```
<标识名 属性 1="值 1" 属性 2="值 2"…/>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

或：

```
<标识名 属性1="值1" 属性2="值2"…>
    <子标识名 属性1="值1" 属性2="值2"…/>
    …        <!-- 多个子标识 -->
</标识名>
```

4.5.1　使用包含动作标识 <jsp:include>

<jsp:include> 动作标识用于包含其他页面，被包含的页面可以是动态页面或静态页面。<jsp:include> 包含的原理是将被包含的页面编译处理后将结果包含在页面中，包含页面的过程如图 4.9 所示。

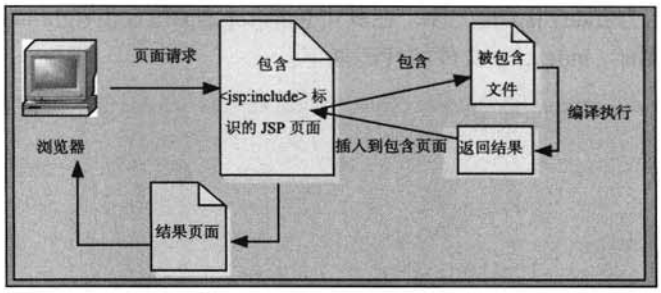


图 4.9　<jsp:include> 动作标识包含页面的过程

当浏览器第 1 次请求一个使用 <jsp:include> 包含其他页面的页面时，Web 容器首先会编译被包含的页面。然后将编译处理后的返回结果包含在页面中，之后编译包含页面，最后将两个页面组合的结果回应给浏览器。

说明　由于静态页面不需要编译即可被 Web 容器解析，所以如果被包含的页面是一个静态页面，将不会经历编译处理这一过程。

<jsp:include> 动作标识的语法格式如下：

```
<jsp:include page="url" flush="false|true" />
```

或：

```
<jsp:include page="url" flush="false|true" >
    子动作标识 <jsp:param>
</jsp:include>
```

- page 属性：指定被包含文件的相对路径。例如，指定属性值为 top.jsp，则表示将与当前 JSP 文件相同文件夹中的 top.jsp 文件包含到当前 JSP 页面中。
- flush 属性：可选，设置是否刷新缓冲区，默认为 false。如果设置为 true，在当前页面输出使用缓冲区的情况下首先刷新缓冲区，然后执行包含操作。

【例 4.5】　应用 <jsp:include> 动作标识包含页面头部页面与版权信息页面。

实例位置：光盘 \MR\ 源码 \ 第 4 章 \4.5

Step 1) 编写一个名为 top.jsp 的文件，用于放置网站的 Banner 信息和导航栏。这里将 Banner 信息和导航栏设计为一张图片。要完成 top.jsp 文件，只需要在该页面通过 标

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

记引入图片即可。top.jsp 文件的代码如下：

```
<%@ page pageEncoding="UTF-8"%>
<img src="images/banner.jpg">
```

Step 2 编写一个名为 copyright.jsp 文件放置网站的版权信息，该文件的代码如下：

```
<%@ page pageEncoding="UTF-8"%>
<table width="780" height="102" border="0" cellpadding="0" cellspacing="0"
background="images/copyright.jpg">
  <tr>
    <td></td>
  </tr>
</table>
```

Step 3 创建一个名为 index.jsp 的文件，在其中包括 top.jsp 和 copyright.jsp 文件，从而实现一个完整的页面。index.jsp 文件的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"pageEncoding="UTF
-8"%>
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>使用 &lt;jsp:include&gt; 动作标识 </title>
</head>
<body style="margin:0px;">
<div align="center">
<jsp:include page="top.jsp"></jsp:include>
<div style="width:780px;height:205px;background-image: url(images/center.jpg)">
</div>
<jsp:include page="copyright.jsp"></jsp:include>
</div>
</body>
</html>
```

运行程序，结果如图 4.10 所示。

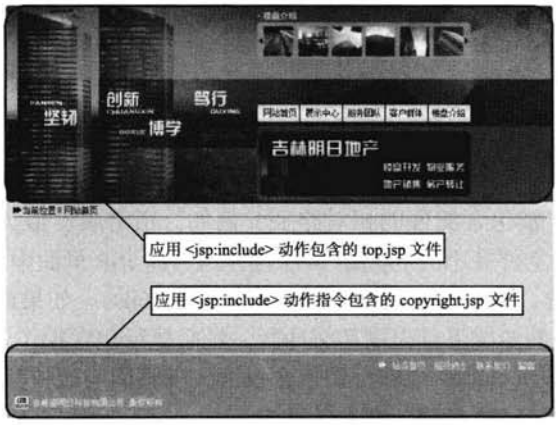


图 4.10　运行结果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明

include 指令与 <jsp;include> 动作标识都可以用来包含文件，二者的差别如下。

1）include 指令通过 file 属性指定被包含的文件，并且该属性不支持任何表达式；<jsp:include> 动作标识通过 page 属性指定被包含的文件，而且该属性支持 JSP 表达式。

2）使用 include 指令时被包含的文件内容会原封不动地插入到包含页中，然后 JSP 编译器将合成后的文件最终编译为一个 Java 文件；使用 <jsp:include> 动作标识包含文件且当该标识被执行时，程序会将请求转发（注意是转发，而不是请求重定向）到被包含的页面并将执行结果输出到浏览器中，然后返回包含页继续执行后面的代码。因为服务器执行多个文件，所以 JSP 编译器会分别编译这些文件。

3）在应用 include 指令包含文件时由于被包含的文件最终会生成一个文件，所以在被包含文件和包含文件中不能有重名的变量或方法；在应用 <jsp:include> 动作标识包含文件时，由于每个文件单独编译，所以在被包含文件和包含文件中重名的变量和方法不冲突。

4.5.2　使用请求转发的动作标识 <jsp:forward>

<jsp:forward> 动作标识将当前请求转发到其他 Web 资源（HTML 页面、JSP 页面和 Servlet 等），在执行请求转发之后当前页面将不再执行，而是执行该标识指定的目标页面。执行请求转发的基本流程如图 4.11 所示。

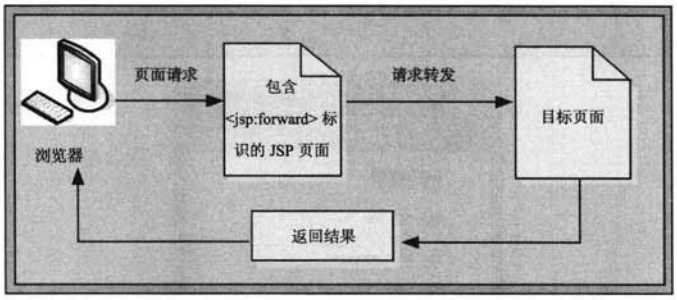


图 4.11　执行请求转发的基本流程

<jsp:forward> 动作标识的语法格式如下：

```
<jsp:forward page="url"/>
```

或：

```
<jsp:forward page="url">
   子动作标识 <jsp:param>
</jsp:forward>
```

page 属性指定请求转发的目标页面，该属性值可以是一个指定文件路径的字符串或表示文件路径的 JSP 表达式。

注意

使用 <jsp:forward> 请求转发被转向到的目标资源必须与当前 JSP 页面处于同一上下文应用环境中，否则会出现页面找不到的错误。

【例 4.6】 应用 <jsp:forward> 动作标识将请求转发到用户登录页面。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

实例位置：光盘 \MR\ 源码 \ 第 4 章 \4.6

Step 1) 编写一个名为 login.jsp 的文件，用于放置用户登录的表单信息，其关键代码如下：

```
<form action="">
<div>
      <ul><li><h3>用户登录</h3></li></ul>
      <ul><li>用户名：<input type="text" name="username"></li></ul>
      <ul><li>密　码：<input type="password" name="password"></li></ul>
      <ul><li><input type="button" value="登　录"></li></ul>
</div>
</form>
```

说明 此页面是请求转发的目标页面，并没有实现用户登录的功能，表单的 action 属性值为空。

Step 2) 创建程序的首页 index.jsp，在其中通过 <jsp:forward> 动作标识将请求转发到 login.jsp 页面，其关键代码如下：

```
<body>
您现在位置为 index.jsp！
<jsp:forward page="login.jsp" ></jsp:forward>
</body>
```

实例运行后，请求会被转发到用户登录页面 login.jsp，运行结果如图 4.12 所示。

说明 请求转发是服务器端的操作，浏览器并不知道请求的页面，所以浏览的地址栏不会发生变化。



图 4.12　运行结果

4.5.3　使用子动作标识 <jsp:param>

JSP 的动作标识 <jsp:param> 可以作为其他标识的子标识，用于为其他标识传递参数，其语法格式如下：

```
<jsp:param name="参数名" value="参数值" />
```

- name 属性：指定参数名称。
- value 属性：设置对应的参数值。

例如，通过 <jsp:param> 标识为 <jsp:forward> 标识指定参数，可以使用下面的代码：

```
<jsp:forward page="update.jsp">
   <jsp:param name="userId" value="1"/>
</jsp:forward>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在上面的代码中实现了在请求转发到 update.jsp 页面的同时传递参数 userId，其值为 1。

说明　通过 <jsp:param> 动作标识指定的参数将以“参数名＝值”形式添加到请求中，其功能与在文件名后面直接加“?参数名＝参数值”相同。

4.5.4　使用动作标识 <jsp:useBean>

使用 <jsp:useBean> 动作标识可以在 JSP 页面中创建一个 Bean 实例，并且通过属性设置可以将该实例保存在 JSP 中的指定范围内。如果在指定的范围内已经存在指定的 Bean 实例，那么将使用这个实例，而不会重新创建。通过 <jsp:useBan> 标识创建的 Bean 实例可以在 Scriptlet 中应用。

该标识的语法格式如下：

```
<jsp:useBean
        id=" 变量名 "
        scope="page|request|session|application"
        {
         class="package.className"|
         type=" 数据类型 "|
         class="package.className" type=" 数据类型 "|
         beanName="package.className" type=" 数据类型 "
        }/>
<jsp:setProperty name=" 变量名 " property="*"/>
```

也可以在标识体内嵌入子标识或其他内容：

```
<jsp:useBean id=" 变量名 " scope="page|request|session|application" >
  <jsp:setProperty name=" 变量名 " property="*"/>
</jsp:useBean>
```

表 4.1 所示为 <jsp:useBean> 标识中各属性的用法。

表 4.1　<jsp:useBean> 标识中各属性用法

属　性	说　明
id	定义一个变量名，程序中将使用该变量名引用所创建的 Bean 实例
type	指定 id 属性所定义变量的类型
scope	定义 Bean 实例的范围，默认值为 page，其他可选值为 rquest、session 和 application
class	指定一个完整的类名，与 beanName 属性不能同时存在。若未设置 type 属性，则必须设置 class 属性
beanName	指定一个完整的类名，与 class 属性不能同时存在。设置该属性时必须设置 type 属性，其属性值可以是一个表示完整类名的表达式

1．id 属性

该属性指定一个变量，在定义的范围内或 Scriptlet 中将使用该变量来引用创建的 Bean 实例。该变量必须符合 Java 中变量的命名规则。

2．type 属性

该属性用于设置由 id 属性指定的变量的类型，它可以指定要创建实例的类本身、类的父

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

类或者是一个接口。

使用 type 属性来设置变量类型的语法格式如下：

```
<jsp:useBean id="us" type="com.Bean.UserInfo" scope="session"/>
```

如果在 session 范围内已经存在名为 us 的实例，则将该实例转换为 type 属性指定的 UserInfo 类型（必须是合法的类型转换）并赋值给 id 属性指定的变量；若指定的实例不存在，将抛出下面的异常：

```
bean us not found within scope
```

上述用法的 <jsp:useBean> 标识被转换为如下对应的 Servlet 代码：

```
com.Bean.UserInfo us = null;      // 定义一个变量，类型由 type 属性指定，变量名由 id 属性指定
synchronized (session) {
   us=(com.Bean.UserInfo)_jspx_page_context.getAttribute("us",PageContext.
   SESSION_ SCOPE);
                                  // 获取 session 范围内名为 "us" 的实例
   if (us == null){               // 如果实例不存在，抛出异常
     throw new java.lang.InstantiationException("bean us not found withinscop e");
    }
}
```

3. scope 属性

该属性指定创建 Bean 实例的存取范围，省略该属性时的值为 page。执行 <jsp:useBean> 标识首先会在 scope 属性指定的范围来查找指定的 Bean 实例，如果该实例已经存在，则引用这个 Bean；否则重新创建，并将其保存在 scope 属性指定的范围内。scope 属性的可选值如下。

1）page：指定创建的 Bean 实例只能在当前的 JSP 文件中使用，包括在通过 include 指令静态包含的页面中有效。

2）request：指定创建的 Bean 实例可以在请求范围内存取，在请求被转发至的目标页面中可通过 request 对象的 getAttribute("id 属性值") 方法获取创建的 Bean 实例。一个请求的生命周期是从客户端向服务器发出一个请求到服务器响应这个请求给用户后结束，请求结束后保存在其中的 Bean 实例失效。

3）session：指定创建 Bean 实例的有效范围为 session，session 是当用户访问 Web 应用时，服务器为用户创建的一个对象，服务器通过 session 的 id 值来区分其他用户。对某一个用户而言，在该范围中的对象可被多个页面共享。

可以使用 session 对象的 getAttribute("id 属性值 ") 方法获取创建的 Bean 实例。

4）application：指定创建的 Bean 实例的有效范围从服务器启动开始到服务器关闭结束。application 对象在服务器启动时创建，被多个用户共享，所以访问该对象的所有用户共享保存在该对象中的 Bean 实例。

可以使用 application 对象的 getAttribute("id 属性值 ") 方法获取创建的 Bean 实例。

4. class 属性

class 属性指定一个完整的类名，其中 package 表示类包的名称；className 表示类的 Class 文件名称。通过 class 属性指定的类不能是抽象的，必须具有公共且没有参数的构造方法。在未设置 type 属性时，必须设置 class 属性。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

使用 class 属性定位一个类的语法格式如下：

```
<jsp:useBean id="us" class="com.Bean.UserInfo" scope="session"/>
```

程序首先会在 session 范围中来查找是否存在名为 us 的 UserInfo 类的实例，如果不存在，则会通过 new 操作符实例化 UserInfo 类来获取一个实例，并以 us 为实例名保存在 session 范围内。上述用法的 <jsp:useBean> 标识被转换为如下对应的 Servlet 代码：

```
com.Bean.UserInfo us = null;
synchronized (session) {
    // 获取 session 范围内名为 "us" 的实例
    us = (com.Bean.UserInfo) _jspx_page_context.getAttribute("us", PageContext.
SESSION_SCOPE);
    if (us == null){                                          // 如果实例不存在
        us = new com.Bean.UserInfo();                         // 创建一个实例
        _jspx_page_context.setAttribute("us", us, PageContext.SESSION_SCOPE);
    }
}
```

说明　代码中的 _jspx_page_context 对象为 JSP 中的 pageContext 内置对象。

5．class="package.className" type=" 数据类型 "

class 与 type 属性可以指定同一个类，在 <jsp:useBean> 标识中使用时的格式如下：

```
<jsp:useBean id="us" class="com.Bean.UserInfo" type="com.Bean.UserBase"scope=
"session"/>
```

这里假设 UserBase 类为 UserInfo 类的父类。该标识被执行时程序首先创建一个以 type 属性值为类型和以 id 属性值为名称的变量 us，并赋值为 null。然后在 session 范围内来查找这个名为 us 的 Bean 实例，如果存在，则将其转换为 type 属性指定的 UserBase 类型（类型转换必须是合法的）并赋值给变量 us；否则通过 new 操作符来实例化一个 UserInfo 类的实例并赋值给变量 us，最后将 us 变量存储在 session 范围内。

上述用法的 <jsp:useBean> 标识被转换为如下对应的 Servlet 代码：

```
com.Bean.UserBase us = null;// 创建一个变量，类型由 type 属性指定，变量名由 id 属性指定
synchronized (session) {
    us = (com.Bean.UserBase) _jspx_page_context.getAttribute("us", PageContext.
    SESSION_SCOPE);
                                                          // 获取 session 范围内名为 "us" 的实例
    if (us == null){                                      // 如果实例不存在
        us = new com.Bean.UserInfo();                     // 实例化一个 UserInfo 类的实例
        _jspx_page_context.setAttribute("us", us, PageContext.SESSION_SCOPE);
    }
}
```

6．beanName="package.className" type=" 数据类型 "

beanName 与 type 属性可以指定同一个类，在 <jsp:useBean> 标识中二者一起使用时的格式如下：

```
<jsp:useBean id="us" beanName="com.Bean.UserInfo" type="com.Bean.UserBase"/>
```

这里假设 UserBase 类为 UserInfo 类的父类。该标识被执行时程序首先创建一个以 type 属

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

性值为类型，以 id 属性值为名称的变量 us，并赋值为 null。然后在 session 范围内来查找这个名为 us 的 Bean 实例，如果存在，则将其转换为 type 属性指定的 UserBase 类型（类型转换必须是合法的）并赋值给变量 us；否则通过 instantiate() 方法从 UserInfo 类中实例化一个类并将其转换为 UserBase 类型后赋值给变量 us，最后将变量 us 存储在 session 范围内。

上述用法的 <jsp:useBean> 标识被转换为如下对应的 Servlet 代码：

```
com.Bean.UserBase us = null;      // 创建一个变量，类型由 type 属性指定，变量名由 id 属性指定
synchronized (session) {
    us = (com.Bean.UserBase) _jspx_page_context.getAttribute("us", PageContext.
    SESSION_SCOPE);
    // 获取 session 范围内名为 us 的实例
    if (us == null){  // 如果实例不存在，通过 instantiate() 方法从 UserInfo 类中实例化一个
        try {
            us = (com.Bean.UserBase) java.beans.Beans.instantiate(this.get Class().
            getClassLoader(), "com.Bean. UserInfo");
        } catch (ClassNotFoundException exc) {
            throw new InstantiationException(exc.getMessage());
        } catch (Exception exc) {
          throw new ServletException("Cannot create bean of class " + "com.
          Bean. UserInfo", exc);
        }
        _jspx_page_context.setAttribute("us", us, PageContext.SESSION_SCOPE);
    }
}
```

通常情况下应用 <jsp:useBean> 标识的如下格式：

```
<jsp:useBean id=" 变量名 " class="package.className"/>
```

如果需要在多个页面中共享这个 Bean 实例，可将 scope 属性设置为“session”。

在页面中使用 <jsp:useBean> 标识来实例化一个 Bean 实例后，可以通过 <jsp:setProperty> 属性来设置或修改该 Bean 中的属性，或者通过 <jsp:getProperty> 标识来读取该 Bean 中指定的属性。

在本节的开始已经介绍了 <jsp:useBean> 标识的两种使用格式。

❑ 不存在 Body 的格式：

```
<jsp:useBean id=" 变量名 " scope="JSP 范围 " .../>                    // 标识结束
<jsp:setProperty name=" 变量名 " property="*"/>
```

❑ 在 Body 内写入内容的格式：

```
<jsp:useBean id=" 变量名 " scope="JSP 范围 ">                         // 标识开始
  <jsp:setProperty name=" 变量名 " property="*"/>
</jsp:useBean>                                                    // 标识结束
```

在页面中应用 <jsp:useBean> 标识创建一个 Bean 时，如果该 Bean 是第 1 次被实例化，那么对于 <jsp:useBean> 标识的第 2 种使用格式，标识体内的内容会被执行；否则不被执行。而对于第 1 种使用格式，无论在指定的范围内是否已经存在一个指定的 Bean 实例，<jsp:useBean> 标识后面的内容都会被执行。

下面通过一个示例来讲解 <jsp:useBean> 标识在程序中的应用。

【例 4.7】 通过 <jsp:useBean> 标识实现一个简单的用户注册程序。在这个实例中，包括

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

两个页面，一个用于显示用户注册表单，另一个用于显示获取到的注册结果。其中，在用户注册页面中，包含一个表单，单击“注册”按钮后表单会被提交到 doreg.jsp 页面，在该页面中应用 <jsp:setProperty> 标识将用户输入的信息赋值给由 <jsp:useBean> 标识所创建的 JavaBean 中的对应属性。若用户输入的信息存在空值，则程序会通过 <jsp:forward> 标识转发到 false.jsp 页面显示注册失败信息；否则通过 <jsp:forward> 标识将请求转发至 success.jsp 页面显示注册成功信息。此时在应用 <jsp:forward> 标识进行转发时，不再需要 <jsp:param> 标识来传递参数。因为在 doreg.jsp 页面中已经将用户输入的信息通过 <jsp:setProperty> 标识保存在由 <jsp:useBean> 标识创建的 JavaBean 实例中，并且该实例保存在 request 请求范围内。所以可以在当前请求被转发到的 success.jsp 页面中来获取这个 JavaBean，从而获得用户输入的注册信息。本示例的实现过程如下。

实例位置：光盘 \MR\ 源码 \ 第 4 章 \4.7

Step 1) 创建一个名为 UserInfo 的 JavaBean，用来保存用户信息，代码如下：

```
package com.Bean;
public class UserInfo {
   private String name;                    // 用户名
   private String job;                     // 职务
   public UserInfo(){
      name="";
      job="";
   }
   public String getName() {
      return name;
   }
   public void setName(String name) throws Exception {
      this.name = new String(name.getBytes("ISO-8859-1"),"UTF-8");
   }
   public String getJob() {
      return job;
   }
   public void setJob(String sex) throws Exception {
      this.job =  new String(sex.getBytes("ISO-8859-1"),"UTF-8");
   }
}
```

说明 JavaBean 是遵循特定规则的 Java 类，关于 JavaBean 请参看第 6 章中的有关内容。

Step 2) 创建提供用户输入注册信息的页面 index.jsp，关键代码如下：

```
<form action="doreg.jsp" method="get" id="form">
<ul>
   <li>用户名: <input type="text" name="name" size="29"></li>
   <li>职　务: <input type="text" name="job" size="29"></li>
   <li><input type="submit" value=" 注册 "><input type="reset" value=" 重置 "></li>
</ul>
</form>
```

由上述代码可得知，form 表单将被提交到 doreg.jsp 页面中，并且表单中两个表单元素的名称 name 和 job 与在 Step 1 中创建的 JavaBean 中的两个属性相同。只有这样设置，才能应用 <jsp:setProperty name="SomeJavaBean" property="*"/> 标识将表单元素与 JavaBean 属性进行

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

一一映射。

Step 3) 创建接收 form 表单的 JSP 页面 doreg.jsp，在其中判断用户输入的信息并将请求转发至指定的页面中，doreg.jsp 页面的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"%>
<jsp:useBean id="us" class="com.Bean.UserInfo" scope="request"/>
<jsp:setProperty name="us" property="*"/>
<%
   String name=us.getName();
   String job=us.getJob();
   if(name.equals("")||job.equals("")){                          // 注册失败
%>
   <jsp:forward page="/false.jsp"/>
<% }else{ %>                                                     // 注册成功
   <jsp:forward page="/success.jsp"/>
<% } %>
```

在上述代码中首先通过 <jsp:useBean> 标识实例化了一个名为 us 且类型为“com.Bean.UserInfo”的 JavaBean 实例，然后保存在 request 请求范围中。<jsp:setProperty> 标识将表单中的元素值赋给这个实例中与表单元素具有相同名字的属性。

Step 4) 创建 false.jsp 页面显示注册失败提示信息，关键代码如下：

```
<ul>
   <li>注册失败！ </li>
   <li>请输入 <b>用户名</b> 或 <b>职务</b>!</li>
   <li><a href="index.jsp">[ 返回 ]</a></li>
</ul>
```

Step 5) 创建 success.jsp 页面显示注册成功提示信息，关键代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"%>
<%@ page import="com.Bean.UserInfo" %>
<%
  String username=((UserInfo)request.getAttribute("us")).getName();
                                                                 // 获取用户名
  String userjob=((UserInfo)request.getAttribute("us")).getJob();
                                                                 // 获取职务
%>
… <!-- 省略了部分 HTML 代码 -->
<ul>
<li>注册成功！ </li>
<li><b>用户名：</b><%=username%>  <b>职    务：</b><%=userjob%>
</li>
   <li><a href="index.jsp">[ 返回 ]</a></li>
</ul>
</div>
</body>
</html>
```

代码中通过 request 对象的 getAttribute() 方法获取保存在 request 请求中的 JavaBean 实例，然后调用实例中的 getXXX() 方法获取属性值。所以最终输出的内容是从 JavaBean 中获取的，而 JavaBean 中的数据是通过 <jsp:useBan> 与 <jsp:setProperty> 标识从 Form 表单中映射而来的。

至此，一个简单的用户注册程序创建完成。运行本实例，输入用户名和职务，如图 4.13

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

所示，单击“注册”按钮后显示如图 4.14 所示的运行结果。



图 4.13　用户注册界面



图 4.14　获取注册结果页面

4.5.5　使用动作标识 <jsp:getProperty>

<jsp:getProperty> 属性用来从指定的 Bean 中读取指定的属性值并输出到页面中，该 Bean 必须具有 getXXX() 方法。

<jsp:getProperty> 标识的语法格式如下：

```
<jsp:getProperty name="Bean 实例名 " property="propertyName"/>
```

- name 属性：指定一个某 JSP 范围中的 Bean 实例。<jsp:getProperty> 标识将会按照 page、request、session 和 application 的顺序来查找这个 Bean 实例，直到第 1 个实例被找到。若任何范围内不存在这个 Bean 实例，则会抛出下面的异常：

```
Attempted a bean operation on a null object
```

- property 属性：指定要获取由 name 属性指定的 Bean 中的哪个属性值。若指定的值为“userName”，那么 Bean 中必须存在 getUserName() 方法，否则会抛出下面的异常：

```
Cannot find any information on property 'userName' in a bean of type '此处为类名'
```

如果指定 Bean 中的属性是一个对象，那么调用该对象的 toString() 方法并输出执行结果。

4.5.6　使用动作标识 <jsp:setProperty>

<jsp:setProperty> 标识通常情况下与 <jsp:useBean> 标识一起使用，它调用 Bean 中的 setXXX() 方法将请求中的参数赋值给由 <jsp:useBean> 标识创建的 JavaBean 中对应的简单属性或索引属性。该标识的语法格式如下：

```
<jsp:setProperty
   name="Bean 实例名 "
    {
          property="*" |
          property="propertyName" |
          property="propertyName" param="parameterName" |
          property="propertyName" value=" 值 "
      }/>
```

<jsp:setProperty> 标识中的属性如下。

（1）name 属性

name 属性指定一个存在 JSP 中某个范围内的 Bean 实例。<jsp:setProperty> 标识将会按照

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

page、request、session 和 application 的顺序来查找这个 Bean 实例，直到找到第 1 个实例。若任何范围内都不存在这个 Bean 实例，则会抛出异常。

（2）property="*"

property 属性值为“*”，则 request 请求中所有参数的值将被一一赋给 Bean 中与参数具有相同名称的属性。如果请求中存在空值的参数，那么 Bean 中对应的属性将不会被赋值为 null；如果 Bean 中存在一个属性，但请求中没有与之对应的参数，那么该属性同样不会被赋值为 null，在这两种情况下的 Bean 属性都会保留原值或默认值。

这种使用方法要求请求中的参数名和类型必须与 Bean 中的属性名和类型一致，但由于通过表单传递的参数都是 String 类型，所以 JSP 会自动将这些参数转换为 Bean 中对应的属性类型。表 4.2 所示为 JSP 将 String 类型转换为其他类型时所调用的方法。

表 4.2　将 String 类型转换为其他类型的方法

其他类型	转换方法	其他类型	转换方法
boolean	java.lang.Boolean.valueOf(String).booleanValue()	Boolean	java.lang.Boolean.valueOf(String)
byte	java.lang.Byte.valueOf(String).byteValue()	Byte	java.lang.Byte.valueOf(String)
double	java.lang.Double.valueOf(String).doubleValue()	Double	java.lang.Double.valueOf(String)
int	java.lang.Integer.valueOf(String).intValue()	Integer	java.lang.Integer.valueOf(String)
float	java.lang.Float.valueOf(String).floatValue();	Float	java.lang.Float.valueOf(String)
long	java.lang.Long.valueOf(String).longValue()	Long	java.lang.Long.valueOf(String)

（3）property="propertyName"

property 属性取值为 Bean 中的属性，则只将 request 请求中与该 Bean 属性同名的一个参数的值赋给这个 Bean 属性。

更进一步讲，如果 property 属性指定的 Bean 属性为 userName，那么指定 Bean 中必须存在 setUserName() 方法，否则会抛出类似下面的异常：

```
Cannot find any information on property 'userName' in a bean of type 'com.Bean.
UserInfo'
```

在此基础上，如果请求中没有与 userName 同名的参数，则该 Bean 属性会保留原值或默认值，而不会被赋值为 null。

当请求中参数的类型与 Bean 中属性类型不一致时，JSP 会自动进行转换。

（4）property="propertyName"，param="parameterName"

param 属性指定一个 request 请求中的参数，property 属性指定 Bean 中的某个属性。这种使用方法允许将请求中的参数赋值给 Bean 中与该参数不同名的属性。如果 param 属性指定的参数值为空，那么由 property 属性指定的 Bean 属性会保留原值或默认值，而不会被赋为 null。

（5）property="propertyName"，value=" 值 "

其中 value 属性指定的值可以是一个字符串数值、表示一个具体值的 JSP 表达式或 EL 表达式，该值将被赋给 property 属性指定的 Bean 属性。

当 value 属性指定一个字符串且指定的 Bean 属性与其类型不一致时，则根据表 4.2 中的方法将该字符串值自动转换为对应的类型。

当 value 属性指定一个表达式时，那么该表达式所表示的值的类型必须与 property 属性指

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

定的 Bean 属性一致，否则会抛出下面的异常：

```
argument type mismatch
```

一般的表单元素在提交表单后，会以“参数名 = 值”的形式添加到请求中，而通过 <input type="image"...> 生成的元素却与此不同。该 HTML 标识将生成一个图片，它被放入表单中后可以用来提交表单，例如：

```
<form action="dosubmit.jsp">
  <input type="image" name="logon" src="image/logon.gif">
  <input type="image" name="reg" src="image/reg.gif">
</form>
```

单击 logon.gif 图片提交表单后，将产生如下请求：

```
dosubmit.jsp?logon.x=50&logon.y=60
```

其中 logon 为 name 属性指定的名称，x 和 y 分别表示鼠标在图片上单击时相对于该图片的 x 和 y 坐标。所以此时请求中的参数并不是表单元素的 name 属性指定的值，而是鼠标单击时的位置。

4.6　上机实践

4.6.1　应用 Java 程序片段动态生成表格

➤➤➤ 题目描述

在实际的项目开发过程中，经常需要通过 JSP 页面动态显示数据库中的数据。如根据条件查询时，需要将查询结果显示在 JSP 页面中，此时需要应用表格来显示查询结果。由于查询结果是动态的，数据行数并不固定，因此在显示查询结果数据时要根据数据的行数来确定表格的行数。本上机实践要求应用 Java 程序片段（Scriptlet）动态生成表格，运行结果如图 4.15 所示。



图 4.15　应用 Java 程序片段动态生成表格

➤➤➤ 技术指导

在 JSP 文件中可以在“<%”和“%>”标记中直接嵌入任何有效的 Java 程序代码，这种嵌入的程序片段称为“Scriptlet”，应用这个标记可以随时在 JSP 中嵌入 Java 程序代码。因此实现本实例非常简单，只要在 JSP 页面的相应位置应用“<%”和“%>”标记即可动态生成表

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

格。本上机实践的关键代码如下：

```
<% String[] bookName = {"Java Web 开发典型模块大全","Java  Web 开发实战宝典" ,"JSP 项目
开发全程实录"};  %>
  <table border="1" align="center">
    <tr>
          <td align="center">编号</td>
          <td align="center">书名</td>
    </tr>
    <%      for(int i=0;i<bookName.length;i++){ %>
    <tr>
          <td align="center"><%=i %></td>
          <td align="center"><%=bookName[i]%></td>
    </tr>
    <%} %>
</table>
```

4.6.2 将页面转发到用户登录页面

➤➤➤ 题目描述

通过 <jsp:forward> 动作标识可以将请求转发到其他 Web 资源，如另一个 JSP 页面、HTML 页面和 Servlet 等。执行请求转发后当前页面将不再被执行，而是执行该标识指定的目标页面。本上机实践要求应用 <jsp:forward> 动作标识将页面转发到用户登录页面，运行结果如图 4.16 所示。



图 4.16　将页面转发到用户登录页面

➤➤➤ 技术指导

本实例主要应用 <jsp:forward> 动作标识的以下语法格式实现。

```
<jsp:forward page="url"/>
```

page 属性指定请求转发的目标页面，该属性值可以是一个指定文件路径的字符串，也可以是表示文件路径的 JSP 表达式。但是请求被转向的目标文件必须是内部的资源，即当前应用中的资源。

本上机实践的关键代码如下：

```
<jsp:forward page="login.jsp"/>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

4.6.3　应用 JavaBean 保存图片信息

➤➤➤ 题目描述

本练习尝试开发一个程序，要求在页面中包含一个 form 表单，并通过 HTML 标记 <input type="image"...> 分别在表单中嵌入两张图片，单击任意图片都可以将表单提交到 dosubmit.jsp 页面。在 dosubmit.jsp 页面中，根据用户的选择将请求转发到指定的页面。运行结果如图 4.17 和图 4.18 所示。

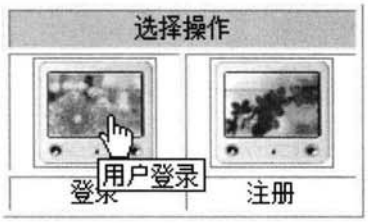


图 4.17　首页运行结果

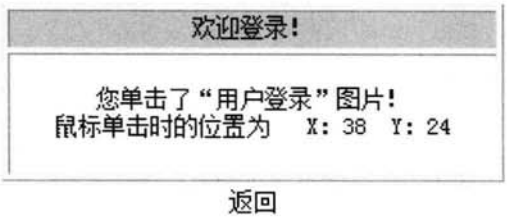


图 4.18　单击“用户登录”图片后的运行结果

➤➤➤ 技术指导

本上机实践主要通过使用 <jsp:forward> 动作标识、<jsp:useBean> 动作标识，以及其子动作标识 <jsp:setProperty> 来实现。在实现该实例时，需要先创建一个保存图片信息的 JavaBean，该 JavaBean 的具体代码如下：

```
public class Image {
   private String x;                                    //x 属性
   private String y;                                    //y 属性
   public Image(){
          x="";                                         // 初始化 x
          y="";                                         // 初始化 y
   }
...       <!-- 此处省略了各属性的 Setter 和 Getter 方法 -->
   //是否被单击方法
   public boolean getSelected(){
          boolean select=false;
          if((this.x!="")&&(this.y!=""))
                 select=true;                           // 表示被单击
          else
                 select=false;                          // 表示没有被单击
          return select;
   }
}
```

然后再应用 <jsp:useBean> 动作标识，及其子标识 <jsp:setProperty> 来保存单击的图片及参数值，关键代码如下：

```
<%@ page language="java" contentType="text/html; charset=gb2312"%>
<%-- 创建一个 JavaBean 实例，对应 " 用户登录 " 图片，并将请求中的参数值赋给 JavaBean 属性 --%>
<jsp:useBean id="logonimg" class="com.javabean.Image" scope="request"/>
<jsp:setProperty name="logonimg" property="x" param="logon.x"/>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<jsp:setProperty name="logonimg" property="y" param="logon.y"/>
<%-- 创建另一个 JavaBean 实例，对应 "用户注册" 图片，并将请求中的参数值赋给 JavaBean 属性 --%>
<jsp:useBean id="regimg" class="com.javabean.Image" scope="request"/>
<jsp:setProperty name="regimg" property="x" param="reg.x"/>
<jsp:setProperty name="regimg" property="y" param="reg.y"/>
<%-- 根据判断结果，将请求转发到指定页面 --%>
<% if(logonimg.getSelected()){ %>
<jsp:forward page="logon.jsp"/>
<% } else if(regimg.getSelected()){ %>
<jsp:forward page="reg.jsp"/>
<% } else{  %>
<jsp:forward page="error.jsp"/>
<% } %>
```

4.7 小结

本章首先介绍了 JSP 页面的基本构成，以及 JSP 的页面指令、文件包含指令和引用标签库指令，然后详细介绍了 JSP 脚本标识和注释，以及常用的 JSP 动作标识。另外，本章还介绍了两种包含文件的方法，即应用 include 指令和 <jsp:include> 动作标识。

4.8 习题

选择题

1. 以下哪个不属于 JSP 指令标签？（　　）

 A. page　　B. include　　C. <jsp:include>

2. 下面哪个不属于 JSP 动作标识？（　　）

 A. <jsp:include>　　B. <jsp:forward>　　C. <c:out>

3. 在 JSP 页面中嵌入 Java 代码需应用哪个标记？（　　）

 A. <% %>　　B. <%-- --%>　　C. <!-- -- >

4. 在 JSP 页面中不包括下面哪项内容？（　　）

 A. JSP 指令标签项　　B. HTML 标记语言　　C. 属性文件

5. 要设置 JSP 页面支持的语言，要设置 page 指令的哪个属性？（　　）

 A. language 属性　　B. extends 属性　　C. contentType 属性

填空题

1. include 指令中的 file 属性的意思是_____。
2. JSP 页面中包含有_____种注释形式。
3. 要包含一个外部文件，需要使用_____动作标识。
4. 要实现将 index.jsp 页面转发至 error.jsp 页面中，需要使用_____标识。
5. 要把 Java 的表达式结果输出到 JSP 页面中，使用的是_____。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第二篇
核心篇

本篇主要内容：

- ❑ 第 5 章　程序开发效率之利器——内置对象
- ❑ 第 6 章　完美的“咖啡豆”——JavaBean 技术
- ❑ 第 7 章　Web 应用的缔造者——Servlet
- ❑ 第 8 章　使用表达式语言（EL）
- ❑ 第 9 章　页面控制利器——JSTL 标签库
- ❑ 第 10 章　数据库应用开发

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 5 章　程序开发效率之利器——内置对象

视频讲解：69 分钟

JSP 提供了由容器实现和管理的内置对象，也可以称为“隐含对象”。这些内置对象不需要 JSP 页面来实例化，在所有的 JSP 页面中都可以直接使用，它起到了简化页面的作用。JSP 的内置对象被广泛应用于 JSP 的各种操作中。

通过学习本章，读者可以达到以下学习目的：

- 掌握应用 request 对象获取请求参数和客户端数据；
- 掌握应用 response 对象设置 HTTP 响应头信息；
- 掌握应用 response 对象实现页面重定向；
- 掌握应用 session 会话对象实现用户登录；
- 掌握应用 application 对象实现网页计数器；
- 了解其他内置对象的应用。

5.1　JSP内置对象概述

为了简化 JSP 开发，JSP 提供了 request、response、session、application、out、page、config、exception 和 pageContext 9 个内置对象。

5.1.1　内置对象的作用

由于 JSP 使用 Java 作为脚本程序语言，所以具有强大的对象处理能力，并且可以动态创建 Web 页面。而在 Java 语言中，需要创建该对象的实例，然后通过引用对象实例，才可以调用一个对象的属性或方法，其操作十分烦琐。JSP 为了简化开发，提供了一些内置对象，用来实现多种 JSP 应用。可以直接使用 JSP 内置对象，在 JSP 页面中内置对象与普通对象调用方式的区别如图 5.1 所示。

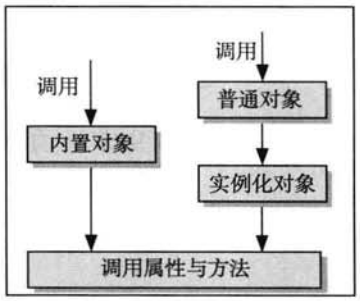


图 5.1　内置对象与普通对象调用方式的区别

5.1.2　内置对象及其应用场合

JSP 内置对象是应用 JSP 进行 Web 开发时，经常被使用的技术。通过它们可以对 Web 开发中的请求、响应、会话、应用、输出、配置信息和异常信息等内容进行控制。对于不同的内容将由不同的内置对象进行控制。下面我们将介绍各种内置对象及其使用场合。

（1）request 对象

该对象封装了客户端提交的 HTTP 请求信息，适用于处理客户端提交的信息，在开发 Web 应用时经常应用 request 对象获取请求参数的值和获取 Cooike 数据等。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（2）response 对象

该对象适用于响应客户端请求信息，在开发 Web 应用时经常应用 response 对象重定向网页、设置 HTTP 响应报头和缓冲区配置等。

（3）session 对象

该对象适用于在同一个应用程序中每个客户端的各个页面中共享数据，session 对象通常应用于保存用户 / 管理员信息和购物车信息等。

（4）application 对象

该对象适用于在同一个应用程序中各个用户间共享数据，application 对象通常应用在计数器或是聊天室中。

（5）out 对象

该对象适用于向客户端输出各种类型的数据，通常用来在 JSP 页面中输出文本。

（6）page 对象

该对象适用于操作 JSP 页面自身，在开发 Web 应用时很少应用。

（7）config 对象

该对象适用于读取服务器的配置信息。

（8）exception 对象

该对象适用于操作 JSP 文件执行时发生的异常信息。

（9）pageContext 对象

该对象适用于获取 JSP 页面的 request、response、session、application 和 out 等对象。由于这些对象均为 JSP 的内置对象，所以在 Web 应用开发时很少使用 pageContext 对象，而是直接使用相应的内置对象。

5.2　request请求对象

request 对象封装了由客户端生成的 HTTP 请求的所有细节，主要包括 HTTP 头信息、请求方式和请求参数等，通过该对象提供的相应方法可以获取客户端请求之中的信息。

5.2.1　获取请求参数

客户端向服务器端发送一个请求时，通常情况下会包含一些请求参数。如请求删除数据库中的一条信息时，需要在请求之中传递一个主键值，在传递的过程中其参数加载在 request 对象中。

当通过超链接的形式发送请求时可以为该请求传递参数，可以通过在超链接的后面加上问号“?”来实现，注意这个问号为英文半角。例如，发送一个请求到 delete.jsp 页面并传递一个名为“id”的参数，可以通过以下超链接实现：

```
<a href="delete.jsp?id=1"> 删除 </a>
```

注意　在通过问号“?”指定请求参数时，参数值不需要使用单引号或双引号括起，包括字符型的参数。如果同时指定多个参数，各参数间使用与符号“&”分隔。

在 delete.jsp 页面中可以通过 request 对象的 getParameter() 方法获取传递的参数值，代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<%
    String id = request.getParameter("id");
%>
```

执行了 getParameter() 方法后变量 id 的值为 1。

说明 在超链接中传递请求参数通过 GET 请求方式传递，如果通过 POST 请求方式传递参数，也使用 getParameter() 方法获取参数值。

【例 5.1】 实现用户注册功能，将用户的注册信息输出到 JSP 页面之中。

实例位置：光盘 \MR\ 源码 \ 第 5 章 \5.1

Step 1) 创建程序首页 index.jsp 的页面，在其中编写用户登录的表单，其关键代码如下：

```
<body>
    <div>
        <form action="login.jsp" method="post">
            <ul>
                <li><h3> 用户登录 </h3></li>
            </ul>
            <ul>
                <li> 用户名: <input type="text" name="username"></li>
            </ul>
            <ul>
                <li> 密  码: <input type="password" name="password"></li>
            </ul>
            <ul>
                <li>
                    <input type="submit" value=" 登  录 ">
                </li>
            </ul>
        </form>
    </div>
</body>
```

Step 2) 创建名为 login.jsp 的用户登录请求的处理页面，其关键代码如下：

```
<%// 获取用户名
    String username = request.getParameter("username");
    // 获取密码
    String password = request.getParameter("password");
    // 判断用户名是否为 mr，密码是否为 mrsoft
    if("mr".equals(username) && "mrsoft".equals(password)){
        // 登录成功
        out.print("<h3> 恭喜，登录成功！ </h3>");
    }else{
        // 登录失败
        out.print("<h3> 对不起，登录失败！ </h3>");
    }%>
```

login.jsp 实现用户登录的功能，其实现方式是判断表单提交的用户名值和密码值是否为 mr 和 mrsoft。如果是，则用户登录成功，否则登录失败。

说明 在使用 request 的 getParameter() 方法获取传递的参数值时，如果指定的参数不存在，则返回 null；如果指定了参数名，但未指定参数值，则返回空的字符串 ""。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

实例运行后用户登录页面的运行结果如图 5.2 所示。在用户登录页面中，输入用户名 mr，密码 mrsoft，单击“登录”按钮，用户登录成功，将显示如图 5.3 所示的运行结果。



图 5.2　用户登录页面的运行结果



图 5.3　用户登录成功

说明　本实例主要讲解 request 对象的应用，并没有涉及数据库。

5.2.2　解决中文乱码问题

在通过 request 对象获取请求参数时，如果参数值为中文且未处理，则获取的参数值将是乱码。在 JSP 中解决获取请求参数中文乱码可以分为以下两种情况。

（1）获取访问请求参数时乱码

当访问请求参数为中文时，通过 request 对象获取的中文参数值为乱码，这是因为该请求参数采用的 ISO-8859-1 编码不支持中文。所以只有将获取的数据通过 String 的构造方法使用 UTF-8 或 GBK 编码重新构造一个 String 对象方可正确显示中文。例如，在获取包括中文信息的参数 user 时可以使用下面的代码：

```
String user = new String(request.getParameter("user").getBytes("iso-8859-1"),
"UTF-8");
```

（2）获取表单提交的信息乱码

当获取表单提交的信息时，通过 request 对象获取的中文参数值为乱码，可以通过在 page 指令的下方加上调用 request 对象的 setCharacterEncoding() 方法将编码设置为 UTF-8 或 GBK 来解决。例如，在获取包括中文信息的用户名文本框（name 属性为 username）的值时，可以在获取全部表单信息前加上下面的代码：

```
<% request.setCharacterEncoding("UTF-8");%>
```

这样通过下面的代码获取表单的值时，就不会产生中文乱码：

```
String user = request.getParameter("username");
```

注意　调用 request 对象的 setCharacterEncoding() 方法的语句时，一定要在页面中没有调用任何 request 对象的方法，否则该语句将不起作用。

5.2.3　获取客户端数据

通过 request 对象可以获取客户端的相关信息，如 HTTP 报头信息、客户信息提交方式，

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

客户端主机 IP 地址及端口号等。在客户端获取用户请求相关信息的 request 对象的常用方法如表 5.1 所示。

表 5.1 request 对象获取客户端信息的常用方法

方 法	说 明
getHeader(String name)	获得 HTTP 协议定义的文件头信息
getHeaders(String name)	返回指定名称的 request Header 的所有值，结果是一个枚举型的实例
getHeadersNames()	返回所有 request Header 的名称，结果是一个枚举型的实例
getMethod()	获得客户端向服务器端传送数据的方法，如 get、post、header 和 trace 等
getProtocol()	获得客户端向服务器端传送数据所依据的协议名称
getRequestURI()	获得发出请求字符串的客户端地址，不包括请求的参数
getRequestURL()	获取发出请求字符串的客户端地址
getRealPath()	返回当前请求文件的绝对路径
getRemoteAddr()	获取客户端的 IP 地址
getRemoteHost()	获取客户端的主机名
getServerName()	获取服务器的名字
getServerPath()	获取客户端所请求的脚本文件的文件路径
getServerPort()	获取服务器的端口号
request.getCookies()	获取客户端保存的 Cookie 数据

【例 5.2】 使用 request 对象的相关方法获取客户端信息。

实例位置：光盘 \MR\ 源码 \ 第 5 章 \5.2

创建 index.jsp 文件，在其中调用 request 对象的相关方法获取客户端信息，该文件的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
… <!-- 此处省略了部分 HTML 代码 -->
<br> 客户提交信息的方式：<%=request.getMethod()%>
<br> 使用的协议：<%=request.getProtocol()%>
<br> 获取发出请求字符串的客户端地址：<%=request.getRequestURI()%>
<br> 获取发出请求字符串的客户端地址：<%=request.getRequestURL()%>
<br> 获取提交数据的客户端 IP 地址：<%=request.getRemoteAddr()%>
<br> 获取服务器端口号：<%=request.getServerPort()%>
<br> 获取服务器的名称：<%=request.getServerName()%>
<br> 获取客户端的主机名：<%=request.getRemoteHost()%>
<br> 获取客户端所请求的脚本文件的文件路径 :<%=request.getServletPath()%>
<br> 获得 HTTP 协议定义的文件头信息 Host 的值 :<%=request.getHeader("host")%>
<br> 获得 HTTP 协议定义的文件头信息 User-Agent 的值 :<%=request.getHeader ("user-agent")%>
<br> 获得 HTTP 协议定义的文件头信息 accept-language 的值 :<%=request.getHeader ("accept-
language")%>
</body>
</html>
```

运行本实例，显示如图 5.4 所示的运行结果。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 5.4　运行结果

说明　默认的情况下，在 Windows 7 系统下，当使用 localhost 进行访问时，应用 request.getRemoteAddr() 获取的客户端 IP 地址将是 0:0:0:0:0:0:0:1，这是以 IPv6 的形式显示的 IP 地址，要显示为 127.0.0.1，需要在 C:\Windows\System32\drivers\etc\hosts 文件中，添加“127.0.0.1 localhost”，并保存该文件。

5.2.4　应用 request 对象域

request 对象可以视为一个域，可以应用 setAttribute() 方法在域范围内存放数据，其语法格式如下：

```
request.setAttribute(String name,Object object);
```

1）name：变量名，为 String 类型，转发后的页面获取数据时通过这个变量名来获取数据。

2）object：指定需要在 request 范围内传递的数据，为 Object 类型。

在将数据保存在 request 范围内的变量中后，可以通过 request 对象的 getAttribute() 方法获取该变量值，其语法格式如下：

```
request.getAttribute(String name);
```

name 为变量名，在 request 范围内有效。

【例 5.3】　应用 request 对象域传递数据。

实例位置：光盘 \MR\ 源码 \ 第 5 章 \5.3

Step 1）创建 index.jsp 文件，在其中首先应用 Java 的 try...catch 语句捕获取页面中的异常信息。如果没有异常，则将运行结果保存在 request 范围内的变量中，并将错误提示信息保存在 request 范围内的变量中，然后应用 <jsp:forward> 动作指令将页面转发到 deal.jsp 页面。index.jsp 文件的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
… <!-- 此处省略了部分 HTML 代码 -->
<%
try{                                                    // 捕获异常信息
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            int money=100;
            int number=0;
            request.setAttribute("result",money/number);              //保存执行结果
}catch(Exception e){
            request.setAttribute("result","很抱歉，页面产生错误！");  //保存错误提示信息
}
%>
<jsp:forward page="deal.jsp"/>
</body>
</html>
```

Step 2 创建 deal.jsp 文件，在其中通过 request 对象的 getAttribute() 方法获取保存在 request 范围内的变量 result 并输出。这里需要注意的是，由于该方法的返回值为 Object 类型，所以需要调用 toString() 方法，将其转换为字符串类型。deal.jsp 文件的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
… <!-- 此处省略了部分 HTML 代码 -->
<%String message=request.getAttribute("result").toString(); %>
<%=message %>
</body>
</html>
```

运行本实例，显示如图 5.5 所示的运行结果。



图 5.5　显示应用 request 对象传递的参数

5.3　response响应对象

response 对象用于响应客户请求并向客户端输出信息，它封装了 JSP 产生的响应，并发送到客户端以响应客户端的请求。请求的数据可以是多种数据类型，甚至是文件。response 对象在 JSP 页面内有效。

5.3.1　操作 HTTP 头信息

通过 response 对象可以设置 HTTP 响应报头，其中常用的是设置响应的内容类型、禁用缓存、设置页面自动刷新和定时跳转网页。

（1）禁用缓存

在默认情况下，浏览器会缓存显示的网页内容。这样当用户再次访问相同的网页时，浏览器会判断网页是否有变化。如果没有变化，则直接显示缓存中的内容，这样可以提高网页的显示速度。对于一些安全性要求较高的网站，通常需要禁用缓存。通过设置 HTTP 头的方法实现禁用缓存可以通过以下代码实现：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<%
    response.setHeader("Cache-Control","no-store");
    response.setDateHeader("Expires",0);
%>
```

（2）设置页面自动刷新

通过设置 HTTP 头可以实现页面的自动刷新，如让网页每隔 10 秒自动刷新一次可以使用下面的代码：

```
<% response.setHeader("refresh","10");%>
```

（3）定时跳转网页

通过设置 HTTP 头可以实现定时跳转网页的功能，如让网页 5 秒钟后自动跳转到指定的页面，可以使用下面的代码：

```
<% response.setHeader("refresh","5;URL=login.jsp");%>
```

5.3.2　设置 MIME 类型

在默认情况下，一个 JSP 页面采用的内容类型是 text/html，即 HTML 或文本数据。此值并不是固定的，可以根据开发需要动态更改响应类型，更改方式是通过 response 对象提供的 setContentType() 方法设置，在设置之后即可让网页内容以不同的格式输出到浏览器中。

setContentType() 方法的语法格式如下：

```
response.setContentType(String type);
```

type 用于指定响应的内容类型，可选值为 text/html、text/plain、application/x_msexcel 和 application/msword 等。

说明　不同的 Web 容器定义 MIME 类型可能存在差异，其中 Tomcat 容器对 MIME 类型的声明定义在 Tomcat 根目录下的“conf\web.xml”文件中，通过 <mime-mapping> 标记声明。

5.3.3　实现页面重定向

使用 response 对象提供的 sendRedirect() 方法可以将网页重定向到另一个页面，重定向操作支持将地址重定向到不同的主机上，这一点与转发不同。在客户端浏览器上将会得到跳转的地址，并重新发送请求链接，用户可以从浏览器的地址栏中看到跳转后的地址。执行重定向操作后 request 中的属性全部失效，并且开始一个新的 request 对象。

sendRedirect() 方法的语法格式如下：

```
response.sendRedirect(String path);
```

path 指定目标路径，可以是相对路径，也可以是不同主机的其他 URL 地址。

例如，使用 sendRedirect() 方法重定向网页到 login.jsp 页面（与当前网页同一主机）和明日编程词典网（与该网页不在同一主机），可以使用下面的代码。

```
response.sendRedirect("login.jsp");                    //重定向到 login.jsp 页面
response.sendRedirect("www.mrbccd.com");               //重定向到明日编程词典网
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

注意 在 JSP 页面中使用该方法时不要有其他 JSP 脚本代码（包括 return 语句），因为重定向之后的代码已经没有意义，并且还可能产生错误。

【例 5.4】 通过 sendRedirect() 方法重定向页面到用户登录页面。

实例位置：光盘 \MR\ 源码 \ 第 5 章 \5.4

Step 1) 创建 index.jsp 文件，在其中调用 response 对象的 sendRedirect() 方法重定向页面到用户登录页面 login.jsp。该文件的关键代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding
="UTF-8"%>
<%response.sendRedirect("login.jsp"); %>
```

Step 2) 编写 login.jsp 文件，在其中添加用于收集用户登录信息的表单及表单元素，关键代码如下：

```
<form name="form1" method="post" action="">
   <p>用户名：<input  name="name" type="text" id="name" style="width: 120px"></p>
   <p>密  码：<input name="pwd" type="password" id="pwd" style="width: 120px"></p>
   <input type="submit" name="Submit" value="提交 ">
</form>
```

运行本实例，默认执行 index.jsp 页面。在其中又执行了重定向页面到 login.jsp 的操作，所以在浏览器中显示如图 5.6 所示的用户登录页面。



图 5.6　用户登录页面

5.4　out输出对象

JSP 通过 out 对象向客户端浏览器输出信息，并且管理应用服务器上的输出缓冲区。在使用该对象输出数据时，可以操作数据缓冲区，并及时清除缓冲区中的残余数据，为其他输出让出缓冲空间。数据输出后要及时关闭输出流。

5.4.1　向客户端输出数据

out 对象的一个基本应用是向客户端浏览器输出信息，该对象可以输出多种数据类型的数据，在输出非字符串类型的数据时会自动转换为字符串输出。out 对象提供了如下两种输出信息的方法。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



（1）print() 方法

print() 方法向客户端浏览器输出信息，通过该方法输出的信息与使用 JSP 表达式输出的信息相同。

例如，下面两行代码均可向客户端浏览器输出文字“明日科技”：

```
<% out.print("明日科技");%>
<%="明日科技" %>
```

（2）println() 方法

println() 方法向客户端浏览器输出信息，并在输出内容后输出一个换行符。

说明 out 内置对象的 print() 和 println() 方法与 Java API 中的 PrintStream（System.out 获取）提供的 print() 和 println() 方法作用相似。

例如，通过 println() 方法向页面中输出字符串“明日科技”及“编程词典”的代码如下：

```
<% out.println("明日科技");
   out.println("编程词典");  %>
```

在上面的代码中，虽然 println() 方法输出了换行“\n”，但在 HTML 语言中输出换行需要使用
 标记，并不会解析“\n”换行符，所以其运行效果如图 5.7 所示。

```
明日科技 编程词典
```

图 5.7　忽略换行符的输出结果

在 JSP 页面中，如果需要使 out 的 println() 方法输出换行，可以使用 HTML 语言提供的 <pre> 标记，如将 <pre> 应用到原来的代码中：

```
<pre>
<% out.println("明日科技");
   out.println("编程词典");%>
</pre>
```

这时，将显示如图 5.8 所示的运行结果。

```
明日科技
编程词典
```

图 5.8　使用 <pre> 后的效果

5.4.2　管理缓冲区

out 对象不仅可以向 JSP 页面中输出内容，而且还可以管理页面中的缓冲区，如清理缓冲区、刷新缓冲区及获取缓冲区大小等。使用对象提供的管理缓冲区的方法如表 5.2 所示。

表 5.2　管理缓冲区的方法

方　法	说　明	方　法	说　明
clear()	清除缓冲区中的内容	flush()	刷新流
clearBuffer()	清除当前缓冲区中的内容	getBufferSize()	获取缓冲区的大小
isAutoFlush()	检测当前缓冲区已满时是自动清空，还是抛出异常		

5.5　session会话对象

由于 HTTP 协议是一种无状态协议，即当一个客户向服务器发出请求时服务器接收请求，并且返回响应后该连接结束，而服务器并不保存相关的信息。为了弥补这一不足，HTTP 协议提供了 session 会话。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

5.5.1 session 的生命周期

session 作用于同一浏览器中，在各个页面中共享数据。无论当前浏览器是否在多个页面间执行了跳转操作，整个用户会话一直存在。直到关闭浏览器，其生命周期如图 5.9 所示。

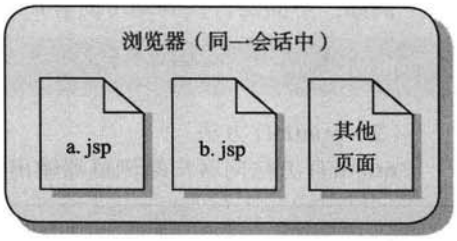


图 5.9 session 生命周期

如果在一个会话中客户端长时间不向服务器发出请求，session 对象就会自动消失，这个时间取决于服务器。例如，Tomcat 服务器默认为 30 分钟，这个时间可以通过编写程序修改。

实际上，一次会话的过程也可以理解为一个打电话的过程。通话从拿起电话或手机拨号开始，一直到挂断电话结束。

5.5.2 创建与获取会话

通过 session 对象可以保存或读取客户相关的信息，如用户名或购物信息等，可以通过 session 对象的 setAttribute() 和 getAttribute() 方法实现。

（1）setAttribute() 方法

该方法将信息保存在 session 范围内，其语法格式如下：

```
session.setAttribute(String name,Object obj)
```

- name：指定作用域在 session 范围内的变量名。
- obj：保存在 session 范围内的对象。

例如，将用户名“无语”保存在 session 范围内的 username 变量中，可以使用下面的代码：

```
session.setAttribute("username","无语");
```

（2）getAttribute() 方法

该方法获取保存在 session 范围内的信息，其语法格式如下：

```
getAtttibute(String name)
```

name 指定保存在 session 范围内的关键字。

例如，读取保存在 session 范围内的 username 变量的值，可以使用下面的代码：

```
session.getAttribute("username");
```

说明 getAttribute() 方法的返回值是 Object 类型，如果需要将获取的信息赋值给 String 类型的变量，需要执行强制类型转换或是调用其 toString() 方法。例如，下面的两行代码都是正确的：

```
String user=(String)session.getAttribute("username");          // 强制类型转换
String user1=session.getAttribute("username").toString();      // 调用 toString() 方法
```

5.5.3 移除对话中的数据

对于保存在 session 会话中的对象，如果需将其从 session 会话中移除，可以使用 session 对象的 removeAttribute() 方法，其语法格式如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
removeAttribute(String name)
```

name 指定作用域在 session 范围内的变量名，一定要保证该变量在 session 范围内有效，否则将抛出异常。

例如，将保存在 session 会话中的 username 对象移除，可以使用下面的代码。

```
<% session.removeAttribute("username");%>
```

5.5.4　设置会话的有效时间

当用户访问网站时会产生一个新的会话，这个会话可以记录用户的状态，但是它并不是永久存在的。如果在一个会话中，客户端长时间不向服务器发出请求，这个会话将被自动销毁。这个时间取决于服务器，如 Tomcat 服务器默认为 30 分钟。不过 session 对象提供了一个设置 session 有效时间的方法 setMaxInactiveInterval()，该方法的语法格式如下：

```
session.setMaxInactiveInterval(int time);
```

time 指定有效时间，单位为秒。如要指定有效时间为一小时，可以指定 time 为 3600。

例如，将 session 的有效时间设置为一小时，可以使用下面的代码：

```
session.setMaxInactiveInterval(3600);
```

说明　在操作 session 时，有时需要获取最后一次与会话相关联的请求时间和两个请求的最大时间间隔，可以通过 session 对象提供的 getLastAccessedTime() 和 getMaxInactiveInterval() 方法实现。其中 getLastAccessedTime() 方法可以返回客户端最后一次与会话相关联的请求时间；getMaxInactiveInterval() 方法返回一个会话内两个请求最大时间间隔，以秒为单位。

虽然当客户端长时间不向服务器发送请求后，session 对象会自动消失。但对于某些实时统计在线人数的网站（如聊天室），每次在 session 过期后才能统计出准确的人数。这是远远不够的，所以还需要手动销毁 session。通过 session 对象的 invalidate() 方法可以销毁 session，其语法格式如下：

```
session.invalidate();
```

session 对象被销毁后，将不可使用。如果在销毁后调用其任何方法，则抛出 Session already invalidated 异常。

5.5.5　应用 session 对象模拟用户登录

通过 session 对象可以保存或读取客户相关的信息，如用户名或购物信息等。下面将通过一个具体的实例介绍应用 session 对象实现用户登录。

【例 5.5】 应用 session 会话对象模拟实现用户登录。

实例位置：光盘 \MR\ 源码 \ 第 5 章 \5.5

Step 1) 创建 index.jsp 文件，在其中添加用于收集用户登录信息的表单及表单元素，其关键代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<form name="form1" method="post" action="">
用户名： <input  name="name" type="text" id="name" style="width: 120px"><br>
密   码： <input name="pwd" type="password" id="pwd" style="width:
120px"> <br>
<br>
<input type="submit" name="Submit" value=" 登录 ">
</form>
```

Step 2) 编写 deal.jsp 文件，在其中模拟用户登录（这里将用户信息保存在一个二维数组中）。如果用户登录成功，将用户名保存在 session 范围内的变量中，并将页面重定向到 main.jsp 页面；否则将页面重定向到 index.jsp 页面，重新登录。deal.jsp 文件的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding
="UTF-8"%>
<%@ page import="java.util.*" %>
<%
String[][] userList={{"mr","mrsoft"},{"wgh","111"},{"sk","111"}};
                                                    //定义一个保存用户列表的二维组
boolean flag=false;                                 //登录状态
request.setCharacterEncoding("UTF-8");              //设置编码
String username=request.getParameter("username");//获取用户名
String pwd=request.getParameter("pwd");             //获取密码
for(int i=0;i<userList.length;i++){                 //遍历二维数组
   if(userList[i][0].equals(username)){             //判断用户名
          if(userList[i][1].equals(pwd)){           //判断密码
                flag=true;                          //表示登录成功
                break;                              //跳出 for 循环
          }
   }
}
if(flag){                                           //如果值为 true，表示登录成功
   session.setAttribute("username",username);       //保存用户名到 session 范围的变量中
   response.sendRedirect("main.jsp");               //跳转到主页
}else{
   response.sendRedirect("index.jsp");              //跳转到用户登录页面
}
%>
```

Step 3) 编写 main.jsp 文件，在其中首先获取并显示保存在 session 范围内的变量，然后添加一个退出超链接。该文件的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding
="UTF-8"%>
<%String username=(String)session.getAttribute("username"); %>
                                                    //获取保存在 session 范围内的用户名
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title> 系统主页 </title>
</head>
<body>
您好！ [<%=username %>] 欢迎您访问！ <br>
<a href="exit.jsp">[ 退出 ]</a>
</body>
</html>
```


Java学习群：72030155

好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Step 4 编写 exit.jsp 文件，在其中销毁 session 并重定向到 index.jsp 页面。该文件的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding
="UTF-8"%>
<%
session.invalidate();                          // 销毁 session
response.sendRedirect("index.jsp");            // 重定向页面到 index.jsp
%>
```

运行本实例，将显示用户登录页面，在该页面中输入用户名 mr，密码 mrsoft，如图 5.10 所示，单击“登录”按钮，将显示如图 5.11 所示的系统主页。



图 5.10　用户登录页面



图 5.11　系统主页

5.6　应用application对象

application 对象用于保存所有应用程序中的公有数据，与 session 对象相比，其生命周期更长，类似系统的“全局变量”。

5.6.1　application 对象的生命周期

application 对象在服务器启动时自动创建，在服务器停止时销毁。当 application 对象没有被销毁时，所有用户都可以共享它，其生命周期如图 5.12 所示。它适用于在同一个应用程序中的各个用户共享数据。

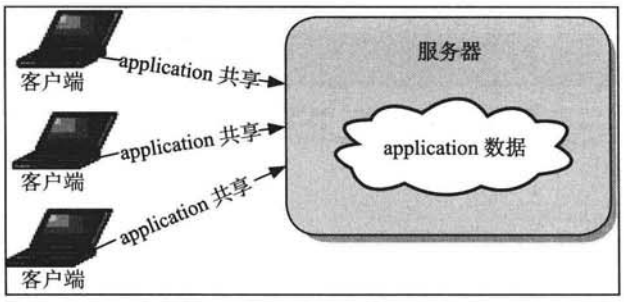


图 5.12　application 对象的生命周期

5.6.2　操作 application 对象中数据

通过 application 对象可以保存、读取或移除应用程序环境属性，如网站访问次数和聊天信息等。应用程序环境属性在 application 范围内有效，管理应用程序环境属性的方法如下。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（1）setAttribute() 方法

setAttribute() 方法用于保存在 application 范围内有效的应用程序环境属性，其语法格式如下：

```
application.setAttribute(String name,Object obj);
```

- name：指定应用程序环境属性的名称。
- obj：属性值，可以是任何 Java 数据类型。

例如，创建一个 application 范围内有效的 number 属性，其属性值为 0，可以使用下面的代码：

```
<%application.setAttribute("number",0); %>
```

（2）getAttributeNames() 方法

getAttributeNames() 方法用于获得所有 application 对象使用的属性名，其语法格式如下：

```
application.getAttributeNames();
```

【例 5.6】获得所有 application 对象的属性名及属性值的代码如下：

实例位置：光盘 \MR\ 源码 \ 第 5 章 \5.6

```
<%@ page import="java.util.*" pageEncoding="UTF-8"%>
<%
application.setAttribute("number",0);
Enumeration enema=application.getAttributeNames();// 获取 application 范围内的全部属性
out.print("<pre>");
while(enema.hasMoreElements()){
   String name=(String)enema.nextElement();          // 获取属性名
   Object value=application.getAttribute(name);      // 获取属性值
   out.print(name+": ");                             // 输出属性名
   out.println(value);                               // 输出属性值
}
out.print("</pre>");
%>
```

上面的代码在运行后，将显示类似图 5.13 所示的信息。

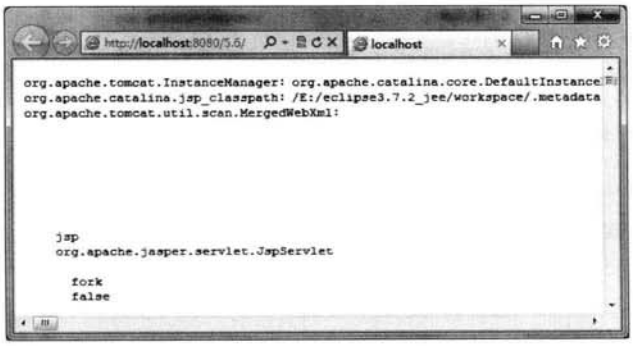


图 5.13　获取的属性值

（3）getAttribute() 方法

getAttribute() 方法用于获取指定属性的属性值，其语法格式如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
application.getAttribute(String name);
```

name 指定属性名，该属性名在 application 范围内有效。

例如，获取 application 范围内的 number 属性的代码如下：

```
<%application.getAttribute("number");%>
```

（4）removeAttribute() 方法

removeAttribute() 方法用于从 application 对象中移除指定名称的属性，其语法格式如下：

```
removeAttribute(String name);
```

name 指定属性名，该属性名在 application 范围内有效，否则将抛出异常。

例如，移出 number 属性的代码如下：

```
<%application.removeAttribute("number");%>
```

5.6.3　配置 Web 应用的初始化参数

application 对象提供了访问应用程序初始化参数的方法，该参数在 web.xml 文件中设置。web.xml 文件位于 Web 应用所在目录下的 WEB-INF 子目录中，在该文件中通过 <context-param> 标记配置应用程序初始化参数。例如，在 web.xml 文件中配置编程词典网的主页地址 home 参数，可以使用下面的代码：

```
<context-param>
   <param-name>home</param-name>
   <param-value>http://www.mrbccd.com</param-value>
</context-param>
```

application 对象提供了如下两种访问应用程序初始化参数的方法。

（1）getInitParameter() 方法

该方法用于返回一个已命名的参数值，其语法格式如下：

```
application.getInitParameter(String name);
```

name 指定参数名。

例如，获取上面 web.xml 文件中配置的 url 参数的值，可以使用下面的代码。

```
application.getInitParameter("url");
```

（2）getAttributeNames() 方法

该方法用于返回所有已定义的应用程序初始化参数名的枚举。

```
application.getAttributeNames();
```

例如，应用 getAttributeNames() 方法获取 web.xml 中定义的全部应用程序初始化参数名，并通过循环输出，可以使用下面的代码。

```
<%@ page import="java.util.*" %>
<%
Enumeration enema=application.getInitParameterNames();  //获取全部初始化参数
while(enema.hasMoreElements()){
   String name=(String)enema.nextElement();             //获取参数名
   String value=application.getInitParameter(name);     //获取参数值
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    out.println(name+": ");                                    //输出参数名
    out.println(value);                                        //输出参数值
}%>
```

如果在 web.xml 文件中，只包括一个上面添加的 home 参数，执行上面的代码显示以下内容：

home：http://www.mrbccd.com

5.6.4 应用 application 对象实现网页计数器

在项目开发中，application 对象常用于实现网页计算器或是聊天室。下面将应用 application 对象实现一个简易的网页计数器。

【例 5.7】 应用 application 对象实现网页计数器的功能。

实例位置：光盘\MR\源码\第 5 章\5.7

创建程序首页 index.jsp 页面，通过内置对象 application 记录来访量，实现网页计数器的功能，其关键代码如下：

```
<body>
<%
        // 获取 application 中存放的 visitorCount 值
    Integer visitorCount = (Integer) application.getAttribute("visitorCount");
        // 判断 visitorCount 是否为空
    if (visitorCount == null) {
                visitorCount = 1;
    }else{
        visitorCount++;          // 来访数量自增
    }
        // 将来访数量保存到 application
    application.setAttribute("visitorCount", visitorCount);
%>
<h3>您是本站的第 <%=visitorCount %> 位访客！</h3>
</body>
```

由于 application 数据在多个客户端间共享，所以实例运行后，无论有多少个客户端访问 index.jsp 页面均计数，运行结果如图 5.14 所示。



图 5.14 网页计数器

5.7 其他内置对象

除了上面介绍的内置对象外，本节简要介绍 JSP 提供的其他内置对象。

5.7.1 应答与请求的 page 对象

page 对象代表 JSP 本身，只在 JSP 页面内才是合法的。该对象本质上是包含当前 Servlet

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

接口引用的变量，可以看做是 this 关键字的别名。page 对象的常用方法如表 5.3 所示。

表 5.3　page 对象的常用方法

方　法	说　明
getClass()	返回当前 Object 的类
hashCode()	返回该 Object 的哈希代码
toString()	把该 Object 类转换为字符串
equals(Object o)	比较该对象和指定的对象是否相等

5.7.2　页面上下文的 pageContext 对象

获取页面上下文的 pageContext 对象是一个比较特殊的对象，通过它可以获取 JSP 页面的 request、response、session、application 及 exception 等对象。该对象的创建和初始化均由容器来完成，在 JSP 页面中可以直接使用 pageContext 对象，其常用方法如表 5.4 所示。

表 5.4　pageContext 对象的常用方法

方　法	说　明
forward(java.lang.String relativeUtlpath)	把页面转发到另一个页面
getAttribute(String name)	获取参数值
getAttributeNamesInScope(int scope)	获取某范围的参数名称的集合，返回值为 java.util.Enumeration 对象
getException()	返回 exception 对象
getRequest()	返回 request 对象
getResponse()	返回 response 对象
getSession()	返回 session 对象
getOut()	返回 out 对象
getApplication	返回 application 对象
setAttribute()	为指定范围内的属性设置属性值
removeAttribute()	删除指定范围内的指定属性

说明　pageContext 对象在实际 JSP 开发过程中很少使用，因为 request 和 response 等对象均为内置对象，并且都可以直接调用其相关方法实现具体的功能，如果通过 pageContext 来调用这些对象，则比较麻烦。

5.7.3　获取 web.xml 配置信息的 config 对象

config 对象主要用于取得服务器的配置信息，通过 pageContext 对象的 getServletConfig() 方法可以获取一个 config 对象。当初始化一个 Servlet 时，容器把某些信息通过 config 对象传递给这个 Servlet。开发人员可以在 web.xml 文件中为应用程序环境中的 servlet 程序和 JSP 页面提供初始化参数。config 对象的常用方法如表 5.5 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 5.5　config 对象的常用方法

方　法	说　明
getServletContext()	获取 Servlet 上下文
getServletName()	获取 Servlet 服务器名
getInitParameter()	获取服务器所有初始参数名称，返回值为 java.util.Enumeration 对象
getInitParameterNames()	获取服务器中 name 参数的初始值

5.7.4　获取异常信息的 exception 对象

exception 对象用来处理 JSP 文件执行时发生的所有错误和异常，只有在 page 指令中设置 isErrorPage 属性值为 true 的页面才可以被使用，在一般的 JSP 页面中使用该对象将无法编译 JSP 文件。该对象几乎定义了所有异常情况，在 Java 程序中可以使用 try...catch 关键字来处理异常情况。如果在 JSP 页面中出现没有捕捉到的异常，则会生成 exception 对象，并将其传送到在 page 指令中设置的错误页面中，然后在错误页面中处理相应的 exception 对象。exception 对象的常用方法如表 5.6 所示。

表 5.6　exception 对象的常用方法

方　法	说　明
getMessage()	返回 exception 对象的异常信息字符串
getLocalizedmessage()	返回本地化的异常错误
toString()	返回关于异常错误的简单信息描述
fillInStackTrace()	重写异常错误的栈执行轨迹

5.8　上机实践

5.8.1　通过 request 对象获取用户注册信息

➤➤➤ 题目描述

通过本章的学习，我们知道通过 request 对象可以获取客户端提交的表单信息，本练习尝试开发一个用户注册程序，要求通过 request 对象获取用户提交的注册信息，效果如图 5.15、图 5.16 所示。输入图 5.15 所示的用户信息并注册，将显示如图 5.16 所示的结果页面。



图 5.15　用户注册页面



图 5.16　显示获取到的注册结果页面

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

➤➤➤ 技术指导

使用 JSP 的 request 内置对象可以获取客户端提交的表单元素的值。通常情况下，要获取文本框、密码、编辑框、隐藏域、单选按钮组和下列表框等表单元素的值时，使用 request 对象的 getParameter() 方法，而要获取复选框和多行列表框等表单元素的值时，则需要使用 request 对象的 getParameterValues() 方法。getParameterValues() 方法的返回值为字符串数组，在显示返回值时，需要使用 for 循环语句遍历该数组。

获取并显示“用户名”文本框的值时，可以使用下面的代码。

```
<input type="text" name="textfield"
          value="<%=new String(request.getParameter("textfield").
getBytes("ISO8859-1"),"UTF-8")%>">
```

获取并显示“爱好”复选框的值时，可以使用下面的代码。

```
   <%
          String[] loves = request.getParameterValues("like");
          for (int i = 0; i < loves.length; i++) {
%>
<%=new String(loves[i].getBytes("ISO8859_1"), "UTF-8")+ "  "%>
<% }       %>
```

5.8.2　防止表单在网站外部提交

➤➤➤ 题目描述

如果静态网页中含有用户提交的表单和字段信息，而从网页的源代码中，又可以看到网页被提交的目标地址，那么修改静态页面表单提交的目标地址，就可以实现在本地运行静态网页并向服务器提交数据的功能。这样，任何人都可以利用网页从外部登录网站，从而给网站留下严重的安全隐患。为此，我们需要解决该问题，本练习尝试应用 JSP 的内置对象来实现防止表单在网站外部提交。实例运行效果如图 5.17 所示。在“用户名”文本框中输入名称，在“密码”文本框中输入密码，单击“提交”按钮即可进入到网页的处理页面，此时地址栏中的地址即为处理页地址（用户也可以通过其他方法获得），当用户在本地计算机上编写静态表单页时，将目标地址设置为以上地址后，可以有效防止表单在网站外部提交。例如，编写一个 HTML 页 aa.html，具体代码如下：

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
<form name="form1" action="http://192.168.1.66:8080/5.sj2/dealwith.jsp"
method="post">
 用户名:  <input type="text" name="name">
 密  码: <input type="password" name="pass">
<input type="submit" name="action2" value=" 提交 ">
<input type="reset" name="Submit" value=" 重置 ">
</form>
</body>
</html>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在浏览器中运行网页并提交表单将显示图 5.18 所示的提示信息。



图 5.17 用户登录页面



图 5.18 从外部提交表单后

➤➤➤ 技术指导

在实现本练习时，首先使用 request.getRequestURL().toString() 方法获得当前网页的地址，然后使用 request.getHeader("referer") 请求页面的地址，接着使用 URL urlOne = new URL(String url) 的方法分别获取两个服务器主机名，最后比较两个主机的名称是否相同。如果网页的 URL 为空，比较时将出现路径有误的信息。本练习的关键代码如下：

```
<%
    String strOne = request.getHeader("referer");          //获取页面的请求地址
    String pathOne = "";                                   //定义空字符串
    if (strOne != null) {                                  //判断当页面的请求地址不为空时
            URL urlOne = new URL(strOne);                  //实例化 URL 方法
            pathOne = urlOne.getHost();                    //获取请求页面的服务器主机
    }
    String strTwo = request.getRequestURL().toString();  //获取当前网页的地址
    String pathTwo = "";
    if (strTwo != null) {
            URL urlTwo = new URL(strTwo);
            pathTwo = urlTwo.getHost();                    //获取当前网页的服务器主机
    }
    if (!pathOne.equals(pathTwo)) {          //判断当前页面的主机与服务器的主机是否相同
            %> 禁止网站外部提交表单！！！ <%
    }
%>
```

5.8.3 通过 cookie 保存并读取用户登录信息

➤➤➤ 题目描述

通过使用 cookie 可以标识用户身份，记录用户名和密码，跟踪重复用户等。浏览器将 cookie 以 key/value 的形式保存到客户机的某个指定目录中。本练习要求尝试通过 cookie 保存并读取用户登录信息。运行结果为第一次显示如图 5.19 所示的页面，输入姓名 mr，并单击“确定”按钮后，将显示如图 5.20 所示的运行结果。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

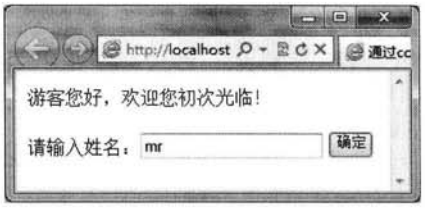


图 5.19　初次运行的结果



图 5.20　登录后再次运行的结果

➤➤➤ 技术指导

通过 cookie 的 getCookies() 方法即可获取到所有 cookie 对象的集合；通过 cookie 对象的 getName() 方法可以获取到指定名称的 cookie；通过 getValue() 方法即可获取到 cookie 对象的值。另外，将一个 cookie 对象发送到客户端，可以使用 response 对象的 addCookie() 方法。

5.9　小结

本章首先简要说明了 JSP 提供的内置对象及其应用场合，然后详细介绍了 request 请求对象、response 响应对象、out 输出对象、session 会话对象和 application 应用对象，这些对象在开发实际项目时经常应用，需要重点掌握。最后简要介绍了 page、pageContext、config 和 exception 对象，这几个内置对象在实际开发中不常用，所以读者不必深入研究。

5.10　习题

选择题

1. 下面哪项内容使用 request 对象无法实现？（　　）
 A. 将页面重定向至 index.jsp 页　　B. 获取表单提交的信息　　C. 定义属性
2. 下面哪项不属于 JSP 内置对象？（　　）
 A. page　　B. session　　C. include　　D. out
3. 获取用户提交的表单属性，使用的是 request 对象的什么方法？（　　）
 A. getParameter() 方法　　B. setCharacterEncoding() 方法　　C. getRemotePort() 方法
4. 设置 session 的有效时间使用的是该对象的什么方法？（　　）
 A. setAttribute() 方法　　B. getMaxInactiveInterval() 方法　　C. setMaxInactiveInterval() 方法
5. session 的有效时间是以什么为单位的？（　　）
 A. 小时　　B. 分钟　　C. 秒

填空题

1. JSP 内置对象包括_____（写出 4 个即可）。
2. 向客户端输出动态内容，需要使用_____内置对象。
3. _____对象表示 JSP 页面本身。
4. 设置所有应用程序中的公有数据，需要使用_____内置对象。
5. 负责处理客户请求的对象是_____。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 6 章　完美的“咖啡豆”——JavaBean 技术

视频讲解：26 分钟

JavaBean 的产生使 JSP 页面中的业务逻辑变得更加清晰，程序之中的实体对象及业务逻辑可以单独封装到 Java 类之中。这样不仅提高了程序的可读性和易维护性，而且还提高了代码的重用性。

通过学习本章，读者可以达到以下学习目的：

- 了解 JavaBean 产生的背景和 JavaBean 的作用；
- 掌握 JavaBean 规范和 JavaBean 的种类；
- 掌握 Eclipse 对 JavaBean 方法的支持；
- 掌握如何获取 JavaBean 属性和对 JavaBean 属性赋值；
- 掌握如何在 JSP 页面中应用 JavaBean。

6.1　JavaBean概述

6.1.1　JavaBean 的产生背景

在 JSP 网页开发的初级阶段并没有框架与逻辑分层概念，需要将 Java 代码嵌入到网页中处理 JSP 页面中的一些业务逻辑，如字符串处理和数据库操作等，其开发流程如图 6.1 所示。

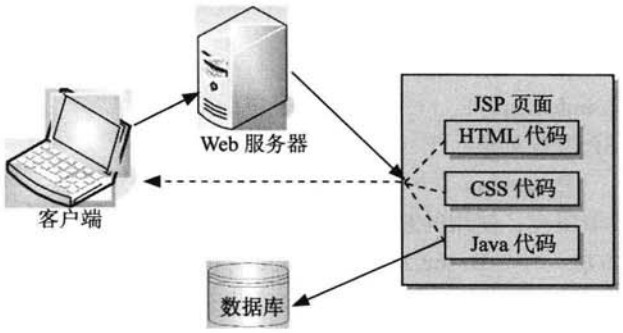


图 6.1　纯 JSP 的开发流程

此种开发流程虽看似简单，但是如果将大量 Java 代码嵌入到 JSP 页面，必定会为修改及维护带来一定的困难。因为在 JSP 页面中包含 HTML、CSS 和 JS 代码等，再添加业务逻辑处理代码，既不利于页面美工人员的设计，也不利于 Java 开发人员开发程序。而且将 Java 代码写入在 JSP 页面中，不能体现面向对象的开发模式，达不到代码的重用。

JavaBean 的产生弥补了原始开发方式的缺点，为 JSP 程序设计注入了新的血液。

6.1.2　JavaBean 的作用

如果使 HTML 与 Java 代码相分离，将 Java 代码单独封装成为一个处理某种业务逻辑的

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

类。然后在 JSP 页面中调用此类，则可以降低 HTML 与 Java 代码之间的耦合度，并且简化 JSP 页面，提高 Java 程序代码的重用性及灵活性。这种与 HTML 代码相分离，而使用 Java 代码封装的类就是一个 JavaBean 组件。在 Java Web 开发可以使用该组件来完成业务逻辑的处理，应用 JavaBean 与 JSP 组合的开发模式如图 6.2 所示。

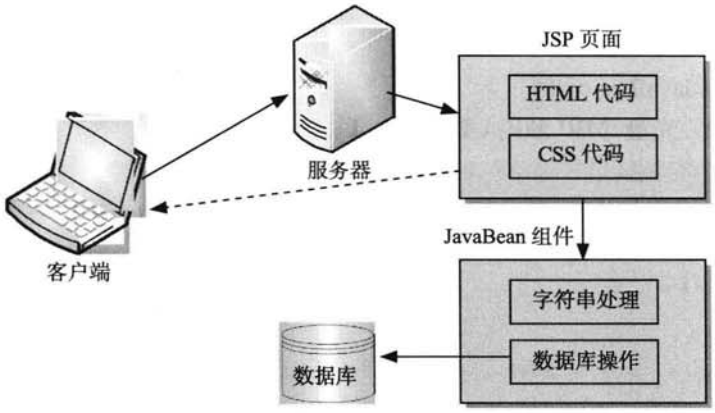


图 6.2　JSP 与 JavaBean 组合的开发方式

可以看出 JavaBean 的应用简化了 JSP 页面，在 JSP 页面中只包含 HTML 和 CSS 代码等，但可以引用 JavaBean 组件来完成某一业务逻辑，如字符串处理和数据库操作等。

6.1.3　JavaBean 的规范

JavaBean 是一个可重用的 Java 软件组件模型，在编写时需要遵循 Java 的编码约定，一个标准的 JavaBean 应该具备以下规范。

- 公共的无参构造方法：一个 JavaBean 对象必须拥有一个公共类型和默认的无参构造方法，从而可以通过 new 关键字直接对其实例化。
- 类的声明是非 final 类型：当一个类声明为 final 类型时，它是不可以更改的，所以 JavaBean 对象的声明应该是非 final 类型。
- 实现可序列接口：JavaBean 应该直接或间接实现 java.io.Serializable 接口，以支持序列化机制。
- 为属性声明访问器：JavaBean 中的属性应该设置为 private（私有）类型，为了防止外部直接访问，需要对外提供公共（public）的访问方法，即需要为属性提供 getter/setter 方法。

说明　上面介绍的 JavaBean 规范只是 JavaBean 应遵循的基本原则，更多的规范约定请参照 JavaBean 规范。

在实际的开发过程中，JavaBean 中的属性应该用小写字母开头，并使用驼峰命名格式命名。

6.1.4　JavaBean 的种类

最初，JavaBean 的研发目的是为了将可以重复使用的代码进行打包，在传统的应用中，JavaBean 主要用于实现一些可视化界面，如一个窗体、按钮和文本框等，这样的 JavaBean 称

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

为可视化的 JavaBean。可视化 JavaBean 一般应用于 Swing 程序中，在 JSP 开发中很少用。

随着技术的不断发展与项目的需求，现在的 JavaBean 主要用于实现一些业务逻辑或封装一些业务对象，由于这样的 JavaBean 并没有可视化的界面，所以又称为非可视化的 JavaBean。非可视化的 JavaBean 又分为值 JavaBean 和工具 JavaBean。值 JavaBean 严格遵循了 JavaBean 的命名规范，通常用来封装表单数据，作为信息的容器。例如，下面的 JavaBean 就是一个值 JavaBean。

【例 6.1】 值 JavaBean 示例。

实例位置：光盘 \MR\ 源码 \ 第 6 章 \6.1

```
public class UserBean{
   private String name;
   private String password;
   public String getName() {
          return name;
   }
   public void setName(String name) {
          this.name = name;
   }
   public String getPassword() {
          return password;
   }
   public void setPassword(String password) {
          this.password = password;
   }
}
```

该 JavaBean 可用来封装用户登录时表单中的用户名和密码。

工具 JavaBean 则可以不遵循 JavaBean 规范，通常用于封装业务逻辑和数据操作等，例如连接数据库，对数据库进行增、删、改、查和解决中文乱码等操作。工具 JavaBean 可以实现业务逻辑与页面显示的分离，提高了代码的可读性与易维护性。例如，下面的 JavaBean 就是一个工具 JavaBean，它用来转换字符串中的“<”与“>”字符。

【例 6.2】 工具 JavaBean 示例。

实例位置：光盘 \MR\ 源码 \ 第 6 章 \6.2

```
public class MyTools{
   public String change(String source){
          source=source.replace("<","&lt;");
          source=source.replace(">","&gt;");
          return source;
   }
}
```

6.2 Eclipse对JavaBean方法的支持

在 JavaBean 属性非常多的情况下，手动编写属性的 getter/setter 方法很不方便。在实际的项目开发过程中，可使用 IDE 开发工具自动生成。下面以 Eclipse 为例介绍如何为属性自动生成 getter 和 setter 方法，其他开发工具大同小异。

【例 6.3】 在 Eclipse 中创建 JavaBean，并自动生成 getter 和 setter 方法，具体操作步骤如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

实例位置：光盘\MR\源码\第6章\6.3

Step 1) 编写一个 JavaBean 对象，并创建相应的属性信息，具体代码如下：

```
public class UserBean {
    private int id;                          //用户 ID
    private String username;                 //用户名
    private String pwd;                      //密码
}
```

Step 2) 在 Eclipse 的代码窗口中，单击鼠标右键，在弹出的快捷菜单中选择“源代码”/“生成 Getter 和 Setter”菜单项，也可以按快捷键〈Alt+Shift+s〉打开该快捷菜单，如图 6.3 所示。

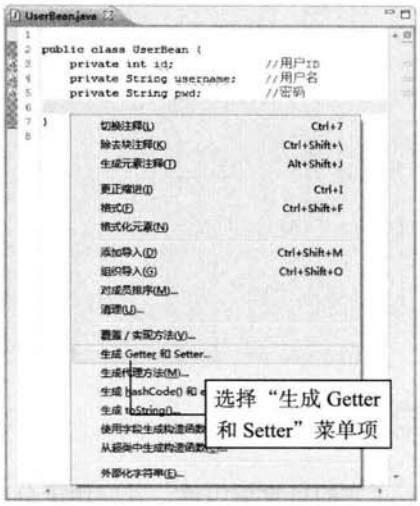


图 6.3　快捷菜单

Step 3) Eclipse 打开“生成 Getter 和 Setter”窗口，单击“全部选中”按钮，选择“访问修饰符”选项组中的“公用”单选按钮，单击“确定”按钮来生成 Getter 和 Setter，如图 6.4 所示。



图 6.4　生成 Getter 和 Setter

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

自动生成的 getter() 与 setter() 方法的代码如下：

```
public class UserBean {
    private int id;                          //用户 ID
    private String username;        //用户名
    private String pwd;                      //密码
    public int getId() {
            return id;
    }
    public void setId(int id) {
            this.id = id;
    }
    public String getUsername() {
            return username;
    }
    public void setUsername(String username) {
            this.username = username;
    }
    public String getPwd() {
            return pwd;
    }
    public void setPwd(String pwd) {
            this.pwd = pwd;
    }

}
```

6.3 JavaBean的应用

JavaBean 是用 Java 语言所写成的可重用组件，其应用十分广泛，可以应用于系统的很多层中，如 PO、VO、DTO 和 POJO 等。

6.3.1 获取 JavaBean 的属性

在 JavaBean 对象中，为了防止从外部直接对 JavaBean 属性的调用，通常将 JavaBean 中的属性设置为私有的（private），但需要为其提供公共的（public）访问方法，也就是所说的 getter 方法。下面就通过实例来讲解如何获取 JavaBean 的属性信息。

【例 6.4】 编写商品对象的 JavaBean。在该 JavBean 中，首先定义相应的属性信息，并为属性提供 getter 方法，然后在 JSP 页面之中获取并输出。

实例位置：光盘 \MR\ 源码 \ 第 6 章 \6.4

Step 1) 编写名称为 Produce 的类，此类是封装商品对象的 JavaBean，在 Produce 类中定义商品属性，并提供相应的 getter 方法，其关键代码如下：

```
package com.wgh;
public class Produce {
    private String name = "编程词典个人版";          // 商品名称
    private double price = 298;    // 商品价格
    private int count = 10;        // 数量
    private String factoryAdd = "吉林省明日科技有限公司";  // 出厂地址
    public String getName() {
            return name;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    }
    public double getPrice() {
            return price;
    }
    public int getCount() {
            return count;
    }
    public String getFactoryAdd() {
            return factoryAdd;
    }
}
```

说明　本实例演示如何获取 JavaBean 中的属性信息，所以对 Produce 类中的属性设置了默认值，可通过 getter 方法直接进行获取。

Step 2) 在 JSP 页面中获取商品 JavaBean 中的属性信息，此操作通过 JSP 动作标识进行获取，其关键代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<jsp:useBean id="produce" class="com.wgh.Produce"></jsp:useBean>
… <!-- 此处省略了部分 HTML 代码 -->
<div>
    <ul>
            <li>
                    商品名称：<jsp:getProperty property="name" name="produce"/>
            </li>
            <li>
                    价格：<jsp:getProperty property="price" name="produce"/>（元）
            </li>
            <li>
                    数量：<jsp:getProperty property="count" name="produce"/>
            </li>
            <li>
                    厂址：<jsp:getProperty property="factoryAdd" name="produce"/>
            </li>
    </ul>
</div>

</body>
</html>
```

说明　在 JSP 网站开发中，JSP 页面中应该尽量避免出现 Java 代码，因为出现这样的代码看起来比较混乱，所以实例中采用 JSP 的动作标识来避免这一问题。

实例中主要通过 <jsp:useBean> 动作标识实例化商品的 JavaBean 对象，<jsp:getProperty> 动作标识获取 JavaBean 之中的属性信息，实例运行后，将显示如图 6.5 所示的运行结果。

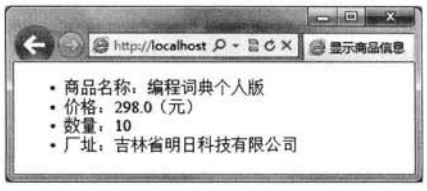


图 6.5　实例运行结果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明 使用 <jsp:useBean> 动作标识可以实例化 JavaBean 对象、<jsp:getProperty> 动作标识可以获取 JavaBean 中的属性信息，这两个动作标识居然可以直接操作我们所编写的 Java 类，是不是在 JSP 页面中可以操作所有的 Java 类呢？答案是否定的。<jsp:useBean> 动作标识与 <jsp:getProperty> 动作标识之所以能够操作 Java 类，是因为我们所编写的 Java 类遵循了 JavaBean 规范。<jsp:useBean> 动作标识获取类的实例，其内部是通过实例化类的默认构造方法进行获取，所以，JavaBean 需要有一个默认的无参的构造方法；<jsp:getProperty> 动作标识获取 JavaBean 中的属性，其内部是通过调用指定属性的 getter 方法进行获取，所以，JavaBean 规范要求为属性提供公共的（public）类型的访问器。只有严格遵循 JavaBean 规范，才能对其进行更好的应用，因此，在编写 JavaBean 时要遵循 Sun 制定的 JavaBean 规范。

6.3.2 对 JavaBean 的属性赋值

编写 JavaBean 对象要遵循 JavaBean 规范，在 JavaBean 规范中的访问器 setter 方法，用于对 JavaBean 中的属性赋值，如果对 JavaBean 对象的属性提供了 setter 方法，在 JSP 页面中就可能通过 <jsp:setProperty> 对其进行赋值。

【例 6.5】 编写封装商品信息的 JavaBean，在这个类中提供属性及与属性相对应的 setter 和 getter 方法，并在 JSP 页面中对 JavaBean 属性赋值并获取输出。

实例位置：光盘 \MR\ 源码 \ 第 6 章 \6.5

Step 1) 编写名称为 Produce 的 JavaBean，用于封装商品信息。在该类中定义商品属性，以及与属性相对应的 setter 和 getter 方法，其关键代码如下：

```
package com.wgh;
public class Produce {
    private String name = "编程词典个人版";                    // 商品名称
    private double price = 298;                                // 商品价格
    private int count = 10;                                    // 数量
    private String factoryAdd = "吉林省明日科技有限公司";       // 出厂地址
    public String getName() {
            return name;
    }
            public void setName(String name) {
            this.name = name;
    }
    …       // 此处省略了其他属性对应的 setter 和 getter 方法
}
```

Step 2) 编写名称为 index.jsp 的页面，在此页面中实例化 Produce 对象，并对其属性进行赋值并输出，其关键代码如下：

```
<jsp:useBean id="produce" class="com.wgh.Produce"></jsp:useBean>
<jsp:setProperty property="name" name="produce" value="手机"/>
<jsp:setProperty property="price" name="produce" value="2980.88"/>
<jsp:setProperty property="count" name="produce" value="1"/>
<jsp:setProperty property="factoryAdd" name="produce" value="深圳 xxx 公司"/>
<div>
    <ul>
            <li>
                    商品名称 :<jsp:getProperty property="name" name="produce"/>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            </li>
            <li>
                价格：<jsp:getProperty property="price" name="produce"/>（元）
            </li>
            <li>
                数量：<jsp:getProperty property="count" name="produce"/>
            </li>
            <li>
                厂址：<jsp:getProperty property="factoryAdd" name="produce"/>
            </li>
    </ul>
</div>
```

index.jsp 页面是程序中的首页，此页面主要通过 <jsp:useBean> 动作标识实例化 Produce 对象，通过 <jsp:setProperty> 动作标识对 Produce 对象中的属性进行赋值，然后再通过 <jsp:getProperty> 动作标识输出已赋值的 Produce 对象中的属性信息，实例运行结果如图 6.6 所示。



图 6.6　对 JavaBean 属性赋值

6.3.3　如何在 JSP 页面中应用 JavaBean

在 JSP 页面中应用 JavaBean 非常简单，主要通过 JSP 动作标识 <jsp:useBean>、<jsp:getProperty>、<jsp:setProperty> 来实现对 JavaBean 对象的操作，但所编写的 JavaBean 对象要遵循 JavaBean 规范，只有严格遵循 JavaBean 规范，在 JSP 页面才能够方便地调用及操作 JavaBean。

将 JavaBean 对象应用到 JSP 页面中，JavaBean 的生命周期可以自行进行设置，它存在于 page、request、session 和 application 4 种范围之内。默认的情况下，JavaBean 作用于 page 范围之内。

【例 6.6】 本实例实现办公自动化系统中员工信息的录入功能，主要通过在 JSP 页面中应用 JavaBean 进行实现。

实例位置：光盘 \MR\ 源码 \ 第 6 章 \6.6

Step 1) 编写名称为 Person 的类，将其放置于 com.wgh 包中，实现对用户信息的封装，其关键代码如下：

```
package com.wgh;
public class Person {
        private String name;                    // 姓名
        private int age;                        // 年龄
        private String sex;                     // 性别
        private String address;                 // 住址
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    …       // 此处省略了其他属性所对应的 getter 和 setter 方法
}
```

在 Person 类中包含 4 个属性，分别代表姓名、年龄、性别与住址，此类在实例中充当员工信息对象的 JavaBean。

Step 2 编写程序的主页面 index.jsp，在此页面中放置录入员工信息所需要的表单，其具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>录入员工信息页面</title>
<style type="text/css">
ul {
   list-style: none;        /*设置不显示项目符号*/
   margin:0px;              /*设置外边距*/
   padding:5px;             /*设置内边距*/
}
li {
   padding:5px;             /*设置内边距*/
}
</style>
</head>
<body>
   <form action="register.jsp" method="post">
      <ul>
         <li>姓　名: <input type="text" name="name"></li>
         <li>年　龄: <input type="text" name="age"></li>
         <li>性　别: <input type="text" name="sex"></li>
         <li>住　址: <input type="text" name="add" size="35"></li>
         <li><input type="submit" value="添　加"></li>
      </ul>
   </form>
</body>
</html>
```

说明 表单信息中的属性名称最好设置成为 JavaBean 中的属性名称，这样就可以通过“<jsp:setProperty property="*"/>”的形式来接收所有参数，此种方式可以减少程序中的代码量。如：将用户年龄文本框的 name 属性设置为“age”，它对应于 Person 类中的 age。

Step 3 编写名称为 register.jsp 的 JSP 页面，用于对 index.jsp 页面中表单的提交请求进行处理，此页面将获取表单提交的所有信息，然后将所获取的员工信息输出到页面之中，

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

其关键代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<% request.setCharacterEncoding("UTF-8");%>
<jsp:useBean id="person" class="com.wgh.Person" scope="page">
   <jsp:setProperty name="person" property="*" />
</jsp:useBean>
… <!-- 此处省略了部分 HTML 和 CSS 代码 -->
<body>
   <ul>
          <li>姓 名：<jsp:getProperty property="name" name="person" /></li>
          <li>年 龄：<jsp:getProperty property="age" name="person" /></li>
          <li>性 别：<jsp:getProperty property="sex" name="person" /></li>
          <li>住 址：<jsp:getProperty property="address" name="person" /></li>
   </ul>
</body>
</html>
```

说明　如果所处理的表单信息中包含中文，通过 JSP 内置对象 request 获取的参数值将出现乱码现象，此时可以通过 request 的 setCharacterEncoding() 方法指定字符编码格式进行解决，实例中将其设置为“UTF-8”。

register.jsp 页面的 <jsp:useBean> 动作标识实例化了 JavaBean，然后通过“<jsp:setProperty name="person" property="*"/>”对 Person 类中的所有属性进行赋值，使用这种方式要求表单中的属性名称与 JavaBean 中的属性名称一致。

说明　表单中的属性名称与 JavaBean 中的属性名称不一致，可以通过 <jsp:setProperty> 动作标识中的 param 属性来指定表单中的属性，如表单中的用户名为“username”，可以使用 <jsp:setProperty name="person" property="name" param="username"/> 对其赋值。

在设置了 Person 的所有属性后，register.jsp 页面通过 <jsp:getProperty> 动作标识来读取 JavaBean 对象 Person 中的属性。实例运行后，将进入到程序的主页面 index.jsp 页面，输入如图 6.7 所示的员工信息后，单击“添加”按钮，将显示如图 6.8 所示的运行结果。



图 6.7　在 index.jsp 页面输入员工信息



图 6.8　register.jsp 页面的运行结果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

6.4 上机实践

6.4.1 判断用户名是否有效

➤➤➤ 题目描述

有些网站在进行用户注册时，用户名只能由字母、数字和下划线组成，并且首字符必须为字母，不允许包含特殊字符等。通过编写工具 JavaBean 就可以验证输入的用户名是否合法。本练习要求尝试编写一个用于判断用户是否有效的 JavaBean，并应用该 JavaBean 判断用户名是否有效，效果如图 6.9 和图 6.10 所示。



图 6.9 输入用户名

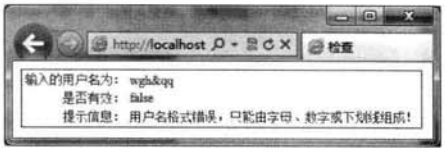


图 6.10 验证结果

➤➤➤ 技术指导

在 JSP 中，可以应用字符的 ASCII 码来判断用户名是否有效，其中英文字母的 ASCII 范围为 65 ~ 90（大写字母）和 97 ~ 122（小写字母）；下划线的 ASCII 码为 95；数字的 ASCII 范围为 47 ~ 58。本上机实践的关键代码参考如下：

```
public boolean isValid() {
        char cArr[] =str.toCharArray();                    //字符串转换为字符数组
        int firstChar =(int) cArr[0];                      //第一个字符的 ASCII 码
        StringBuffer sb = new StringBuffer("");
        //判断首字符是否为字母
        if((firstChar>=65&&firstChar<=90)||(firstChar>=97&&firstChar<=122)){
                for(int i=1;i<cArr.length;i++){
                        int ascii =cArr[i];                //获得字符的 ASCII 码
                        //判断字符是否为字母、数字或下划线，下划线的 ASCII 码为 95
                        if((ascii>=48&&ascii<=57)||(ascii>=65&&ascii<=90)||
                        (ascii==95)||(ascii>=97&&ascii<=122)){
                                sb.append(cArr[i]);
                                //如果条件满足，将字符添加到 StringBuffer 字符串的末尾
                        }
                }
                int length = cArr.length-sb.toString().length();
              //如果被判断字符串长度与 StringBuffer 字符串记录的长度差 1(即去掉首字符的长度)
                if(length==1){
                        this.setCue("用户名格式正确！");
                        valid=true;
                        return valid;
                }else{
                        this.setCue("用户名格式错误，只能由字母、数字或下划线组成！");
                        valid=false;
                        return valid;
                }
        }else{                              //如果首字符不是字母，直接返回 false
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                this.setCue("用户名格式不对，首字符必须为字母！");
                valid=false;
                return valid;
            }
        }
```

6.4.2　计算两个日期相差的天数

➤➤➤ 题目描述

在实际开发过程中有时会遇到计算两个日期相差天数的情况，这也可以通过编写一个工具 JavaBean 来实现。本练习要求尝试编写一个用于计算两个日期相差天数的 JavaBean，并应用该 JavaBean 计算输入的两个日期之间相差的天数，效果如图 6.11 和图 6.12 所示。

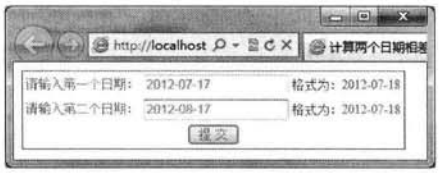


图 6.11　输入日期

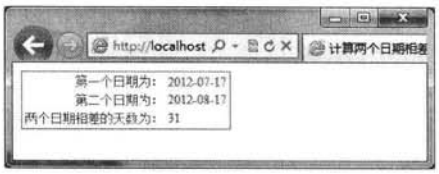


图 6.12　计算结果

➤➤➤ 技术指导

本实例主要通过使用 Calendar 对象中的 getTimeInMillis() 方法，返回一个 long 型的时间值，以毫秒为单位。首先获得两个日期对象的 long 类型的时间值，这两个值相减之后除以一天的毫秒值（1000×60×60×24），得到的值取整即两个日期相差的天数。本上机实践的关键代码参考如下：

```
    public int getMinus() {
        Calendar c1 = this.getCalendar(dateStr1);
                    //根据第一个日期字符串获得 Calendar 对象
        Calendar c2 = this.getCalendar(dateStr2);
                    //根据第二个日期字符串获得 Calendar 对象
        long t1 = c1.getTimeInMillis();
                    //获得此对象的时间值，以毫秒为单位
        long t2 = c2.getTimeInMillis();
                    //获得此对象的时间值，以毫秒为单位
        long t = 1000*60*60*24;
                    //1000 毫秒 *60 秒 *60 分钟 *24 小时
        minus =  (int)((t2-t1)/t);
                    //获得两个日期相差的天数
        return minus;
    }
```

6.4.3　随机生成指定位数的验证码

➤➤➤ 题目描述

验证码技术是网站开发过程中保护网站安全的最基本环节，它可以防止非法人员利用注册工具或登录工具来攻击网站（也就是常说的灌水），从而保护网站安全。本练习要求编写一个

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

随机生成指定位数验证码的程序。效果如图 6.13 和图 6.14 所示。

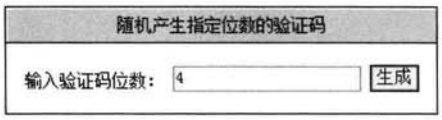


图 6.13　输入生成验证码的位数

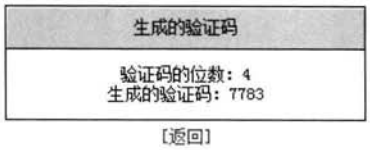


图 6.14　生成结果

➤➤➤ 技术指导

首先设置一个字符串，该字符串中包含了 0 到 9（包括 0 和 9）的数字。将该字符串的长度减 1，然后从 0 到该长度的范围内随机取出一个数字，作为从该字符串中截取长度为 1 的子字符串的起始位置。该范围内的随机数可通过下面的方法得到：int pos=((int)Math.random()*100)%9。之后就可以使用 substring(pos,pos+1) 方法来截取字符串了。本上机实践的关键代码如下：

```
public void makeChecknum(){
        String sourcenum="0123456789";          //定义获取随机数的源字符串
        String siglenum="";                     //保存获取到的单个随机数
        String checknum="";                     //获取到的随机数
        int index=0;                            //获取随机数的位置
        int i=0;
        while(i<this.number){
                index=((int)(Math.random()*100))%(sourcenum.length()-1);
                //随机生成获取随机数的位置
                siglenum=sourcenum.substring(index,index+1);
                //获取单个随机数
                checknum+=siglenum;
                //连接获取到的随机数
                i++;
        }
        this.checknum=checknum;
}
```

6.5　小结

本章首先介绍了 JavaBean 对象的产生背景及使用 JavaBean 的好处，然后介绍了 JavaBean 规范和种类。其中 JavaBean 的规范需要重点掌握，因为只有严格遵循 JavaBean 规范的 JavaBean 对象在 JSP 页面之中才能被很好地引用，否则在操作 JavaBean 的过程中程序将出现不正确的结果或异常。最后介绍了如何获取 JavaBean 的属性、如何对 JavaBean 属性赋值和如何在 JSP 页面中应用 JavaBean，这部分内容需要读者熟练掌握。

6.6　习题

选择题

1．下面哪个属于 JavaBean 文件？（　　）

A. User.java　　B. user.html　　C. user.jsp

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2．在 JSP 中应用 JavaBean 使用的是（　　）标记。

A. <jsp:userBean>　　B. <jsp:setProperty>　　C. <jsp:getProperty>

3．设置 JavaBean 属性，使用的是 (　　) 标记。

A. <jsp:userBean>　　B. <jsp:setProperty>　　C. <jsp:getProperty>

4．获取 JavaBean 属性值，使用的是（　　）标记。

A. <jsp:userBean>　　B. <jsp:setProperty>　　C. <jsp:getProperty>

5．<jsp:useBean> 标签的 scope 属性，不可以设置为（　　）。

A. request　　B. session　　C. out

填空题

1．在纯 JSP 开发模式中，JSP 页面中可以包含有_____。

2．在 JavaBean 规范中，JavaBean 的属性需要设置为_____。

3．JavaBean 的作用域可以设置为_____。

4．<jsp:useBean> 标签中的 type 属性的含义是_____。

5．<jsp:useBean> 标签中应用_____属性设置应用的类名。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 7 章　Web 应用的缔造者——Servlet

视频讲解：46 分钟

在 Java Web 程序开发中，Servlet（Java 服务器小程序）主要用于处理各种业务逻辑，它比 JSP 更具有业务逻辑层的意义。而且其安全性、扩展性及性能都十分优秀，在 Java Web 程序开发及 MVC 模式的应用方面具有极其重要的作用。

通过学习本章，读者可以达到以下学习目的：

- 理解 Servlet 的生命周期及其核心 API；
- 学会应用 Servlet 实现页面转发；
- 理解 Servlet 过滤器的工作原理；
- 学会应用 Servlet 过滤器；
- 理解 Servlet 监听器的原理；
- 掌握应用 Servlet 监听器查看在线用户。

7.1　Servlet技术

Servlet 先于 Java EE 平台出现，在过去的一段时间内曾经得到过广泛应用，如今在 Java EE 项目开发中仍然被广泛使用，是一种非常成熟的技术。

7.1.1　Servlet 概述

在 JSP 开发过程中，人们不断地将 JSP 进行模式化的分离处理，从而很好地分离了网页中的表示、业务处理及逻辑处理层，增强了程序的可扩展性及可维护性。最初的 JSP 开发模式为 JSP+JavaBean，即 Model 1 模式。在建立中小型的网站中，这种模式应用较多。JSP+Servlet+JavaBean 则慢慢演变成了 Model 2 模式，它在实际中得到更为广泛的应用，一般大型网站中均采用这种技术构建。

Servlet 是使用 Java Servlet 应用程序设计接口（API）及相关类和方法的 Java 程序，除了 Java Servlet API，它还可以用于扩展和添加 API 的 Java 类软件包。Java 语言能够实现的功能，Servlet 基本上都能实现（除了图形界面外）。Servlet 主要用于处理客户端传来的 HTTP 请求，并返回一个响应。通常所说的 Servlet 就是指 HttpServlet，能够处理的请求有 doGet()、doPost() 和 service() 等方法。在开发 Servlet 时，可以直接继承 javax.servlet.http.HttpServlet。

7.1.2　Servlet 功能

Servlet 通过创建一个框架来扩展服务器的能力，以在 Web 上提供请求和响应服务。当客户端发送请求至服务器时，服务器可以将请求信息发送给 Servlet，并让其建立服务器返回给客户端的响应。当启动 Web 服务器或客户端第一次请求服务时，可以自动装入 Servlet，之后 Servlet 继续运行直到其他客户端发出请求。Servlet 的工作流程如图 7.1 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

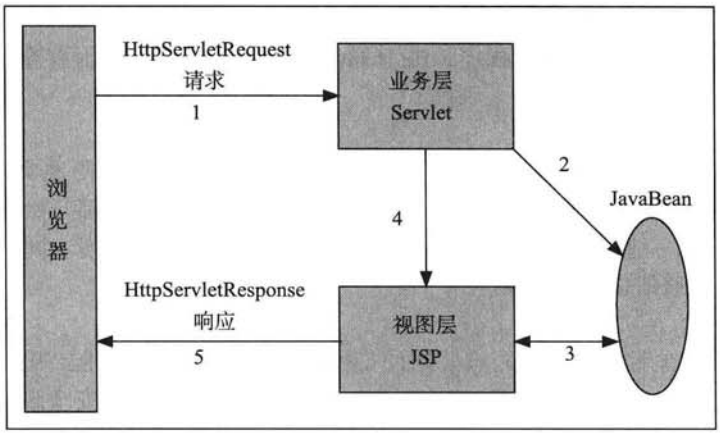


图 7.1　Servlet 的工作流程

Servlet 的主要功能如下。

1）创建并返回一个包含基于客户请求性质动态内容的完整 HTML 页面。

2）创建可嵌入到现有 HTML 页面中的部分 HTML 页面（HTML 片段）。

3）与其他服务器资源（包括数据库和基于 Java 的应用程序）通信。

4）用多个客户端处理连接，接收多个客户端的输入，并将结果传递到多个客户端上。例如，Servlet 可以是多参与者的游戏服务器。

5）当允许在单连接方式下传送数据的情况下，在浏览器上打开服务器至 Applet 的新连接，并将该连接保持在打开状态。当允许客户端和服务器简单且高效地执行会话的情况下，Applet 也可以启动客户浏览器和服务器之间的连接，可以通过定制协议来通信。

6）将定制的处理提供给所有服务器的标准程序，如 Servlet 可以修改如何认证用户。

7.1.3　Servlet 特点

Servlet 技术带给开发人员最大的优势是可以处理客户端传来的 HTTP 请求，并返回一个响应。总的来说，Servlet 技术具有以下特点。

（1）高效

在服务器上仅有一个 Java 虚拟机在运行，其优势在于当多个来自客户端的请求访问时，Servlet 为每个请求分配一个线程，而不是进程。

（2）方便

Servlet 提供了大量的实用工具例程，如处理很难完成的 HTML 表单数据、读取和设置 HTTP 头，以及处理 Cookie 和跟踪会话等。

（3）跨平台

Servlet 用 Java 类编写，可以在不同操作系统平台和不同应用服务器平台下运行。

（4）功能强大

在 Servlet 中，许多使用传统 CGI 程序很难完成的任务都可以轻松地完成。例如，Servlet 能够直接和 Web 服务器交互，而普通的 CGI 程序不能。Servlet 还能够在各个程序之间共享数据，使得数据库连接池之类的功能很容易实现。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（5）灵活性和可扩展性

采用 Servlet 开发的 Web 应用程序，由于 Java 类的继承性及构造函数等特点，使得应用灵活，可随意扩展。

（6）共享数据

Servlet 之间可通过共享数据容易地实现数据库连接池，并能方便地实现管理用户请求，简化 Session 和获取前一页面信息的操作。由于每个 CGI 程序的调用都开始一个新的进程，在 CGI 之间通信则很差，调用间通信通常要通过文件实现，因而相当缓慢。同一台服务器上的不同 CGI 程序之间的通信也相当麻烦。

（7）安全

有些 CGI 版本有明显的安全弱点。即使是使用最新的标准和 PERL 等语言系统也没有基本的安全框架。而 Java 定义有完整的安全机制，包括 SSL\CA 认证和安全政策等规范。

7.1.4 Servlet 的生命周期

Servlet 部署在容器中，生命周期由容器管理，可概括为以下几个阶段。

1）当 Web 客户请求 Servlet 服务或当 Web 服务启动时，容器环境加载一个 Java Servlet 类。

2）容器环境将根据客户请求创建一个或者多个 Servlet 对象实例，并把这些实例加入到 Servlet 实例池中。

3）容器环境调用 Servlet 的初始化方法 HttpServlet.init() 完成初始化，这需要为 init() 方法传入一个 ServletConfig 对象。其中包含初始化参数和容器环境的信息，并负责向 Servlet 传递数据。如果传递失败，则会发生 ServletException 异常，Servlet 将不能正常工作。

4）容器环境利用一个 HttpServletRequest 和 HttpServletResponse 对象封装从 Web 客户接收到的 HTTP 请求和由 Servlet 生成的响应。

5）容器环境把 HttpServletRequest 和 HttpServletResponse 对象传递给 HttpServlet.service() 方法，这样一个定制的 Java Servlet 即可访问这种 HTTP 请求和响应接口。service() 方法可被多次调用，各调用过程运行在不同的线程中，互不干扰。

6）定制的 Java Servlet 从 HttpServletRequest 对象读取 HTTP 请求数据，访问来自 HttpSession 或 Cookie 对象的状态信息，执行特定应用的处理并且用 HttpServletResponse 对象生成 HTTP 响应数据。

7）当 Web 服务器和容器关闭时会自动调用 HttpServlet. destroy() 方法关闭所有打开的资源，并执行一些关闭前的处理。

Servlet 生命周期的各个阶段如图 7.2 所示。

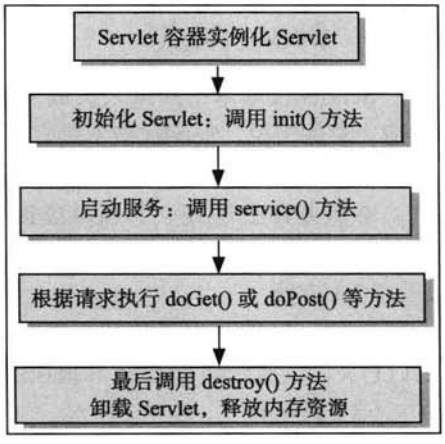


图 7.2 Servlet 生命周期各个阶段

7.2 Servlet技术开发

7.2.1 Servlet 在 Java EE 中的结构体系

Servlet 是一个标准，定义在 Java EE API 中，其具体细节由 Servlet 容器实现，如 Tomcat

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

和 JBoss 等。在 Java EE 架构中，Servlet 结构体系的 UML 如图 7.3 所示。

在图中 Servlet、ServletConfig 与 Serializable 是接口，其中 Serializable 是 java.io 包中的序列化接口，Servlet 和 ServletConfig 是 javax.servlet 包中定义的接口，这两个接口定义了 Servlet 的基本方法并封装了 Servlet 的相关配置信息。GenericServlet 是一个抽象类，分别实现上述 3 个接口。此接口为 Servlet 及 ServletConfig 接口提供了部分实现，但并没有实现 Http 请求处理。这一操作由其子类 HttpServlet 实现，这个类为 Http 请求中 POST、GET 等类型提供了具体的操作方法。

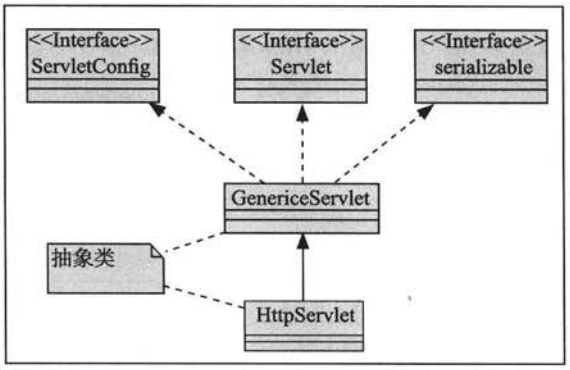


图 7.3　Servlet 结构体系的 UML

通常情况下，我们编写的 Servlet 类都继承于 HttpServlet，在开发中使用的具体的 Servlet 对象就是 HttpServlet 对象。这是因为 HttpServlet 为 Servlet 做出实现，并提供了 Http 请求的处理方法。

7.2.2　Servlet 核心 API

学习 Servlet 需要掌握 Servlet API 中定义的常用接口与类，在 Servlet 开发中会经常用到这些对象及相关方法，Servlet 的常用 API 如下。

（1）Servlet 接口

Servlet 接口位于 javax.servlet 包中，它定义了 Servlet 的主要方法，方法声明及其说明如表 7.1 所示。

表 7.1　Servlet 接口的常用方法声明及其说明

方法声明	说　明
public void service(ServletRequest request,ServletResponse response)	Servlet 在处理客户端的请求时调用此方法
public void destroy()	当 Servlet 容器移除 Servlet 对象时调用此方法，以释放资源
public ServletConfig getServletConfig()	用于获取 Servlet 对象的配置信息，返回 ServletConfig 对象
public String getServletInfo()	返回有关 Servlet 的信息，它是纯文本格式的字符串，如作者和版本等

（2）HttpServlet 类

HttpServlet 类是 Servlet 接口的实现类，主要封装了处理 Http 请求的方法，常用方法声明及其说明如表 7.2 所示。

表 7.2　HttpServlet 类的常用方法声明及其说明

方法声明	说　明
protected void doGet(HttpServletRequest req, HttpServletResponse resp)	用于处理 GET 类型的 Http 请求的处理方法
protected void doPost(HttpServletRequest req, HttpServletResponse resp)	用于处理 POST 类型的 Http 请求的方法
protected void doPut(HttpServletRequest req, HttpServletResponse resp)	用于处理 PUT 类型的 Http 请求的方法

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（3）HttpServletRequest 接口

HttpServletRequest 接口位于 javax.servlet.http 包中，它封装了 Http 的请求。通过此接口可以获取客户端传递的 Http 请求参数，常用方法声明及其说明如表 7.3 所示。

表 7.3 HttpServletRequest 接口的常用方法声明及其说明

方法声明	说明
public String getContextPath()	返回上下文路径，此路径以“/”开头
public Cookie[] getCookies()	返回所有 Cookie 对象，返回值类型为 Cookie 数组
public String getMethod()	返回 HTTP 请求的类型，如 GET 和 POST 等
public String getQueryString()	返回请求的查询字符串
public String getRequestURI()	返回主机名到请求参数之间的部分字符串
public HttpSession getSession()	返回与客户端关联的 HttpSession 对象

（4）HttpServletResponse 接口

HttpServletResponse 接口位于 javax.servlet.http 包中，它封装了对 Http 请求的响应。通过此接口可以向客户端发送回应，其常用方法声明及说明如表 7.4 所示。

表 7.4 HttpServletResponse 接口的常用方法声明及其说明

方法声明	说明
public void addCookie(Cookie cookie)	向客户端发送 Cookie 信息
public void sendError(int sc)	发送一个错误状态码为 sc 的错误响应到客户端
public void sendError(int sc, String msg)	发送包含错误状态码及错误信息的响应到客户端
public void sendRedirect(String location)	将客户端请求重定向到新的 URL

说明 本节仅介绍了 Servlet API 中的常用接口与类，更多的方法及声明请参阅 Java EE API 文档。

7.2.3 创建第一个 Servlet 类

由于 HttpServlet 类是针对 HTTP 请求处理封装的 Servlet 类，能够处理 Java Web 开发中浏览器发送的 HTTP 请求，所以通过继承 HttpServlet 类来编写一个 Servlet 类是最合适不过的方法。在继承 HttpServlet 类后，即可通过重写 HttpServlet 中的方法来编写具体的业务实现。

【例 7.1】 创建一个名称为 MyServlet 的 Servlet 类的具体代码如下：

实例位置：光盘 \MR\ 源码 \ 第 7 章 \7.1

```
import java.io.IOException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class MyServlet extends HttpServlet {
    public void init(ServletConfig config) throws ServletException {
        // 业务处理
    }
    public void destroy() {
        // 业务处理
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
            // 业务处理
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
            // 业务处理
    }
    protected void doPut(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
            // 业务处理
    }
    ...
}
```

说明　HttpServlet 类提供了丰富的 HTTP 请求处理方法，除了 init() 方法与 destroy() 方法外，还有很多针对不同 HTTP 请求类型提供的方法，在此不一一列举。

在 HttpServlet 类提供的多个请求处理方法中，对于不同的 HTTP 请求类型，Web 容器将自动按请求类型分发处理。如 GET 类型的请求将分发给 doGet() 方法处理，POST 类型的请求分发给 doPost() 方法处理，如图 7.4 所示。

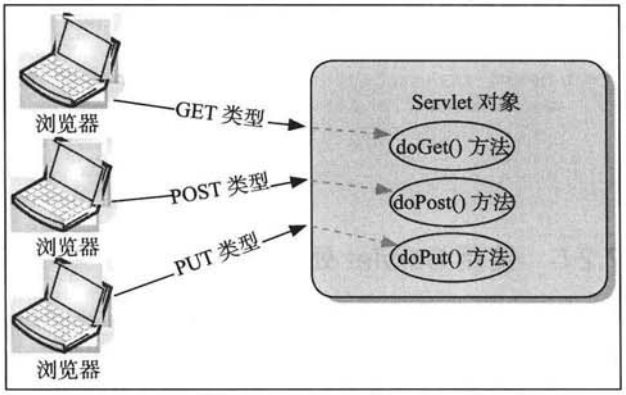


图 7.4　请求分发方式

由于 Web 容器会自动按请求类型调用 Servlet 提供的方法，所以在实际应用过程中，根据请求类型提供相应的方法即可。如一个 Servlet 只处理 GET 类型的请求，那么只需要编写重写 doGet() 方法，不需要在继承 HttpServlet 后实现每个请求方法。

说明　虽然 HttpServlet 对象提供了多个请求处理方法，但在实际的开发中，doGet() 与 doPost() 方法应用最为频繁，因为在 B/S 结构中主要的请求类型是 GET 与 POST。

7.2.4　Servlet 配置

创建了 Servlet 类后，还需要对 Servlet 进行配置，配置的目的是为了将创建的 Servlet 注册到 Servlet 容器之中，以方便 Servlet 容器对 Servlet 的调用。在 Servlet 3.0 以前的版本中，只能在 web.xml 文件中配置 Servlet，而在 Servlet 3.0 中，除了可以在 web.xml 文件中配置 Servlet 以外，还提供了利用注解来配置 Servlet 的方法。下面将分别介绍这两种方法。

1．在 web.xml 文件中配置 Servlet

【例 7.2】　在 web.xml 文件中配置 com.mingrisoft 包中的 MyServlet 的具体代码如下：

实例位置：光盘 \MR\ 源码 \ 第 7 章 \7.2

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<!-- 注册 Servlet -->
<servlet>
        <!-- Servlet 描述信息 -->
        <description>This is my first Servlet</description>
        <!-- Servlet 的名称 -->
        <servlet-name>MyServlet</servlet-name>
        <!-- Servlet 类的完整类名 -->
        <servlet-class>com.mingrisoft.MyServlet</servlet-class>
</servlet>
<!-- Servlet 映射 -->
<servlet-mapping>
        <!-- Servlet 名称 -->
        <servlet-name>MyServlet</servlet-name>
        <!-- 访问 URL 地址 -->
        <url-pattern>/MyServlet</url-pattern>
</servlet-mapping>
```

2. 采用注解配置 Servlet

【例 7.3】 采用注解的方式配置 com.mingrisoft 包中的 MyServlet 的具体代码如下：

实例位置：光盘 \MR\ 源码 \ 第 7 章 \7.3

```
import javax.servlet.annotation.WebServlet;
@WebServlet("/MyServlet")
public class MyServlet extends HttpServlet {
    ...
}
```

7.2.5 使用 Servlet 处理业务逻辑

本节通过一个实例讲解如何使用 Servlet 处理业务逻辑。

【例 7.4】 通过 Servlet 处理表单提交的请求。

实例位置：光盘 \MR\ 源码 \ 第 7 章 \7.4

Step 1) 创建并通过注解配置名称为 DealServlet 的 Servlet 类，用于处理浏览器请求，此类继承于 HttpServlet 类并重写了 doPost() 方法，其关键代码如下：

```
package com.wgh.servlet;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/DealServlet")
public class DealServlet extends HttpServlet {
    // 处理 POST 请求
    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
    response)
     */
    protected void doPost(HttpServletRequest request,
                    HttpServletResponse response) throws ServletException,
                    IOException {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            request.setCharacterEncoding("UTF-8");
            response.setContentType("text/html; charset=UTF-8");
            // 获取 name 参数值
            String name = request.getParameter("name");
            // 从 response 中获取 PrintWriter 对象
            PrintWriter writer = response.getWriter();
            // 判断 name 是否为空
            if (name != null) {
                    // 如果 name 不为空，则通过 writer 输出 name
                    writer.write("Hello " + name + " !");
            }
            writer.flush(); // 刷新 writer
            writer.close(); // 关闭 writer
        }
    }
```

说明　从 HttpServletResponse 对象中获取的输出流，在完成业务逻辑处理后，应刷新缓冲区的数据并关闭输出流对象，以及时释放资源。

Step 2) 创建名为 index.jsp 的页面，作为程序的主页面。在其中编写一个表单，用于向 Servlet 提交请求，其关键代码如下：

```
<form action="HelloServlet" method="post">
    <p>请输入您的姓名：<input type="text" name="name"></p>
    <p><input type="submit" value="提　交"/></p>
</form>
```

实例的运行结果如图 7.5 所示，在表单中输入数据并单击“提交”按钮后的运行结果如图 7.6 所示。



图 7.5　运行结果

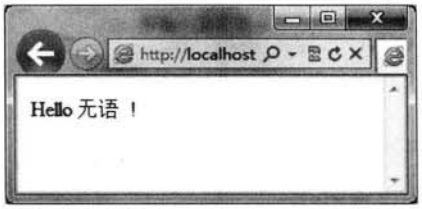


图 7.6　单击“提交”按钮后的运行结果

说明　该实例处理 Http 请求类型中的 POST 请求，如果处理 GET 类型或其类型的请求，在 Servlet 中需要调用与请求类型相对应的处理方法。

7.2.6　使用 Eclipse 快速创建 Servlet

使用 IDE 集成开发工具创建 Servlet 比较简单，适合于初学者。本节以 Eclipse 开发工具为例，介绍创建方法。

Step 1) 创建一个动态 Web 项目，然后在包资源管理器中，在新建项目名称节点上，单击鼠标右键，在弹出的快捷菜单中，选择“新建”/Servlet 菜单项，将打开 Create Servlet 对话框，在该对话框的 Java package 文本框中输入包 com.mingrisoft，在 Class name 文本框中输入类名 FirstServlet，其他的采用默认设置，单击“下一步”按钮，如图 7.7 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

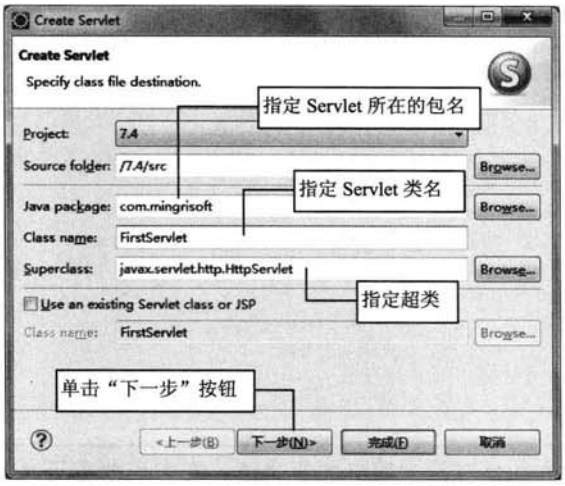


图 7.7　Create Servlet 对话框

Step 2) 进入到如图 7.8 所示的指定配置 Servlet 部署描述信息页面，在该页面中采用默认设置。



图 7.8　配置 Servlet 部署描述的信息

说明　在 Servlet 开发中，如果需要配置 Servlet 的相关信息，可以在图 7.8 中所示的窗口中进行配置，如描述信息、初始化参数、URL 映射。其中，“描述信息”指对 Servlet 添加了一段描述文字；“初始化参数”指在 Servlet 初始化过程中用到的参数，这些参数可以利用 Servlet 的 init 方法进行调用；“URL 映射”指通过哪一个 URL 来访问 Servlet。

Step 3) 单击“下一步”按钮，将进入到如图 7.9 所示的用于选择修饰符、实现接口和要生成

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

的方法的对话框。在该对话框中，修饰符和接口保持默认，在“继承的抽象方法”复选框中选中 doGet 和 doPost 复选框，单击“完成”按钮，完成 Servlet 的创建。

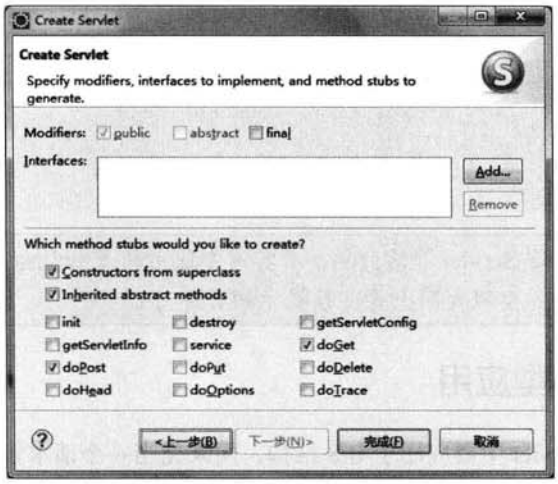


图 7.9　选择修饰符、实现接口和生成的方法

说明　选择 doPost 与 doGet 复选框的作用是让 Eclipse 自动生成 doGet() 与 doPost() 方法，实际应用中可以选择多个方法。

Servlet 创建完成后，Eclipse 将自动打开该文件。创建的 Servlet 类的代码如下：

```
package com.mingrisoft;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * Servlet 实现类 FirstServlet
 */
@WebServlet("/FirstServlet")
public class FirstServlet extends HttpServlet {
   private static final long serialVersionUID = 1L;
    /**
     * @see HttpServlet#HttpServlet()
     * 构造方法
     */
    public FirstServlet() {
        super();
    }
    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
   response)
     */
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            // 业务处理
    }
    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
    response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
            // 业务处理
    }
}
```

说明 使用开发工具创建 Servlet 非常简单，本实例中使用的是 Eclipse IDE for Java EE 工具。其他开发工具操作步骤大同小异，按提示操作即可。

7.3 Servlet的典型应用

在实际开发中，Servlet 主要应用于 B/S 结构，用来充当一个请求控制处理的角色。当浏览器发送一个请求时，由 Servlet 接受并对其执行相应的业务逻辑处理，最后对浏览器做出回应。

7.3.1 处理表单数据

【例 7.5】 通过 Servlet 处理表单数据，实现添加用户信息的功能，并将信息放到 ServletContext 中，然后通过 JSP 页面查看。

实例位置：光盘 \MR\ 源码 \ 第 7 章 \7.5

Step 1 创建名为 User 的类，它是封装用户信息的 JavaBean，关键代码如下：

```
package com.wgh;
public class UserBean {
    private String name;                    // 用户姓名
    private String sex;                     // 性别
    private String address;                 // 家庭住址
    public String getName() {
            return name;
    }
    public void setName(String name) {
            this.name = name;
    }
    // 此处省略了其他属性所对应的 Getter 和 Setter 方法
}
```

Step 2 创建 index.jsp 页面，它是程序的首页。主要用于放置收集人员信息的表单，其关键代码如下：

```
    <form action="AddressServlet" method="post" onsubmit="return check(this);">
            <h2>添加人员信息 </h2>
            <ul>
                    <li>姓 名：<input type="text" name="name"></li>
                    <li>性 别：  <input type="radio" name="sex" value="男"
                            checked="checked">男 <input type="radio" name="sex"
value="女">女
                    </li>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                <li>家庭住址：<textarea rows="5" cols="30" name="add">
                </textarea></li>
                <li><input type="submit" value="添　加"></li>
            </ul>
    </form>
```

为了防止客户端提交空的用户信息，需要验证表单信息。实例中通过 JavaScript 脚本验证，其关键代码如下：

```
<script type="text/javascript">
    function check(form){
        with(form){
            if(name.value == ""){
                alert("姓名不能为空！");
                return false;
            }
            if(add.value == ""){
                alert("家庭住址不能为空！");
                return false;
            }
        }
    }
</script>
```

Step 3 创建处理表单数据的 Servlet 对象 AddressServlet，在其中重写超类 HttpServlet 对象中的 doPost() 方法，其关键代码如下：

```
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");          // 设置 request 的编码格式
        String name = request.getParameter("name");     // 获取用户姓名
        String sex = request.getParameter("sex");       // 获取性别
        String address = request.getParameter("address"); // 获取家庭住址
        UserBean user = new UserBean();                 // 实例化 User
        user.setName(name);                             // 为姓名赋值
        user.setSex(sex);                               // 为性别赋值
        user.setAddress(address);                       // 为家庭住址赋值
        ServletContext application = getServletContext();
                                        // 获取 ServletContext 对象
        List<UserBean> list = (List<UserBean>) application.
        getAttribute("users");          // 从 ServletContext 中获取 users
        if(list == null){               // 判断 List 是否为 null
            list = new ArrayList<UserBean>();           // 实例化 list
        }
        list.add(user);                 // 将 user 添加到 List 集合中
        application.setAttribute("users", list);
                                        // 将 List 放置于 Application 范围中
        RequestDispatcher dispatcher = request.getRequestDispatcher("manager.
        jsp");                          // 创建 RequestDispatcher 对象
        dispatcher.forward(request, response);   // 请求转发到 manager. jsp 页面
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明 HttpServletRequest 对象的 setCharactorEncoding() 方法用于设置请求中的字符编码，实例中在 doPost() 方法中的第 1 行代码中调用。如果未设置 HttpServletRequest 对象的字符编码，处理中文时将出现乱码现象。

Step 4 创建名为 manager.jsp 的 JSP 页面，用于显示输出所添加的人员数据，其关键代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@page import="java.util.List"%>
<%@page import="com.wgh.UserBean"%>
…          <!-- 省略了部分 HTML 和 CSS 代码 -->
<h2 style="width:350px;text-align: center;">用户信息列表 </h2>
   <table align="center" border="1" width="400">
          <tr align="center" style="font-weight: bold;">
                 <td> 姓名 </td>
                 <td> 性别 </td>
                 <td> 家庭住址 </td>
          </tr>
          <%
                 List<UserBean> list = (List<UserBean>)application.
                 getAttribute("users");
                 if(list != null){
                        for(UserBean user : list){
          %>
                 <tr align="center">
                        <td><%=user.getName()%></td>
                        <td><%=user.getSex()%></td>
                        <td><%=user.getAddress()%></td>
                 </tr>
          <%
                        }
                 }
          %>
          <tr>
                 <td align="center" colspan="3">
                        <a href="index.jsp"> 继续添加 </a>
                 </td>
          </tr>
   </table>
```

在 index.jsp 页面中主要通过 for 循环输出迭代用户信息集合中的数据，其人员信息集合从内置对象 Application 中获取。

运行本实例，将显示添加用户信息的页面，在该页面中输入用户信息后，如图 7.10 所示，单击“添加”按钮，表单数据将提交给 AddressServlet 处理，处理后将请求转发到 manager.jsp 页面回应。实例中的用户信息可以多次添加，添加的信息将被放置到 List 集合中并保存在 ServletContext 中。添加多个用户信息后的运行结果如图 7.11 所示。



图 7.10 添加人员信息的表单



图 7.11 添加多个人员信息后的运行结果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

7.3.2　页面转发

在 Servlet 中实现页面转发主要是利用 RequestDispatcher 接口实现的。RequestDispatcher 接口可以把一个请求转发到另一个 JSP 页面。该接口包括以下两个方法。

（1）forward() 方法

forward() 方法用于把请求转发到服务器上的另一个资源，可以是 Servlet、JSP 或是 HTML。forward() 方法的语法格式如下：

```
requestDispatcher.forward(HttpServletRequest request,HttpServletResponse
response)
```

其中，requestDispatcher 为 RequestDispatcher 对象的实例。

（2）include() 方法

include() 方法用于把服务器上的另一个资源（Servlet、JSP、HTML）包含到响应中。include() 方法的语法格式如下：

```
requestDispatcher.include(HttpServletRequest request,HttpServletResponse
response)
```

其中，requestDispatcher 为 RequestDispatcher 对象的实例。

【例 7.6】 编写一个 Servlet 程序，在网站运行时，将页面直接跳转到网站首页 main.jsp 页面。

实例位置：光盘 \MR\ 源码 \ 第 7 章 \7.6

Step 1) 创建名称为 ForwardServlet.java 类文件，该类继承了 HttpServlet 类。在该 Servlet 的 doGet() 方法中调用 RequestDispatcher 接口的 forward() 方法，将页面转发到 main.jsp 页面。ForwardServlet 类的具体代码如下：

```
package com.wgh;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * Servlet 实现类 ForwardServlet
 */
@WebServlet("/ForwardServlet")    // 通过注解配置 ForwardServlet
public class ForwardServlet extends HttpServlet {
   private static final long serialVersionUID = 1L;
    public ForwardServlet() {
        super();
    }
    /**
     * 执行 HTTP GET 请求
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        RequestDispatcher requestDispatcher=request.getRequestDispatcher("main.
        jsp");                    // 创建 RequestDispatcher 类的对象
        requestDispatcher.forward(request, response);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            // 转发页面
        }
    }
```

在上面的代码中，通过注解来配置 Servlet，配置后的 URI 映射为 /ForwardServlet。

Step 2 打开 IE 浏览器，在地址栏中输入地址“http://localhost:8080/7.6/ForwardServlet”，则会出现如图 7.12 所示的运行结果。



图 7.12　通过 Servlet 实现页面转发

7.3.3　获取当前页的绝对路径

Servlet/JSP 程序需要操作服务器所拥有的资源，为此需要得到其绝对路径或实际路径，然后执行下一步操作。

【例 7.7】 在 Servlet 中获取当前服务器的路径。

实例位置：光盘 \MR\ 源码 \ 第 7 章 \7.7

创建 SurePath.java 类文件，该类继承了 HttpServlet。主要通过 doGet() 方法编写获取路径的代码，其代码如下：

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html; charset=UTF-8");
    ServletContext context = this.getServletContext();
    PrintWriter out = response.getWriter();
    out.print("获取当前服务器的真实路径：" + context.getRealPath("") + "");
}
```

在上述代码中，应用了 ServletContext 类，该类定义了一组方法用于 Servlet 容器之间的通信，以获得相关信息，如文件的 MIME 类型、分发请求和日志等。ServletContext 类的 getRealPath() 方法用于返回指定虚拟路径的绝对路径，它的语法格式如下：

```
String getRealPath(String path);
```

本实例的运行结果如图 7.13 所示。

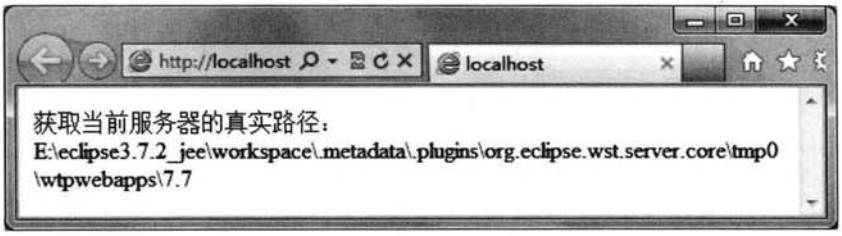


图 7.13　运行结果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

7.4　配置并使用Servlet过滤器

Java Web 中的 Servlet 过滤器主要用于过滤客户端（浏览器）的请求，然后将过滤后的请求转交给下一资源，它在 Java Web 开发中具有十分重要的作用。

7.4.1　过滤器的处理方式

Servlet 过滤器与 Servlet 十分相似，但具有拦截客户端（浏览器）请求的功能，可以改变请求中的内容来满足实际开发的需要。对于开发人员而言，过滤器实质就是在 Web 应用服务器上的一个 Web 应用组件，用于拦截客户端（浏览器）与目标资源的请求，并执行一定过滤处理后发送给目标资源，过滤器的处理方式如图 7.14 所示。

在 Web 容器中部署了过滤器以后，不仅客户端发送的请求会经过过滤器的处理，而且请求在发送到目标资源处理以后其回应信息也同样要经过过滤器。

如果一个 Web 应用中使用一个过滤器不能满足业务需要，那么可以部署多个过滤器进行多次处理。这样组成了一个过滤器链，Web 容器按过滤器的先后顺序处理请求，如图 7.15 所示。

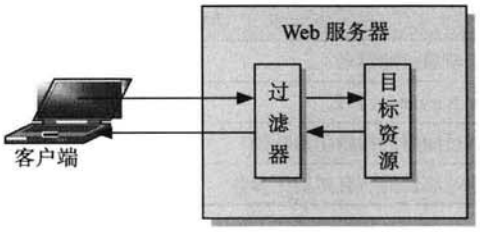


图 7.14　过滤器的处理方式

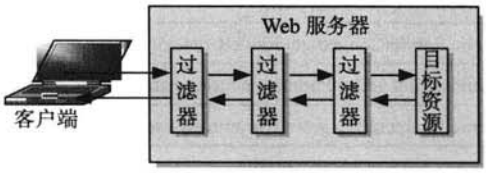


图 7.15　过滤器链

在第 1 个过滤器处理一个请求后，会传递给第 2 个过滤器处理，直到最后一个过滤器将请求交给目标资源处理。目标资源在处理经过过滤的请求后，其回应信息从最后一个过滤器依次传递到第 1 个过滤器，最后传送到客户端，这就是过滤器在过滤器链中的应用流程。

7.4.2　过滤器的核心对象

过滤器对象放置在 javax.servlet 包中，其名为 Filter，它是一个接口。除这个接口外，与过滤器相关的对象还有 FilterConfig 与 FilterChain 对象，这两个对象也是接口对象，位于 javax.servlet 包中，分别为过滤器的配置对象与过滤器的传递工具。在实际开发中，定义过滤器对象只需要直接或间接地实现 Filter 接口。图 7.16 所示为 MyFilter1 与 MyFilter2 过滤器，其中 FilterConfig 与 FilterChain 对象用于执行过滤器的相关操作。

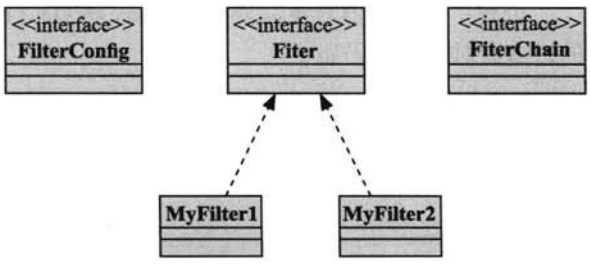


图 7.16　MyFilter1 与 MyFilter2 过滤器

（1）Filter 接口

每一个过滤器对象都要直接或间接地实现 Filter 接口，在该接口中定义了 3 个方法，即

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

init()、doFilter() 与 destroy()，其方法声明及说明如表 7.5 所示。

表 7.5 Filter 接口的方法声明及说明

方法声明	说明
public void init(FilterConfig filterConfig) throws ServletException	过滤器初始化方法，此方法在初始化过滤器时调用
public void doFilter (ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException	对请求进行过滤处理
public void destroy()	销毁方法以释放资源

（2）FilterConfig 接口

FilterConfig 接口由 Servlet 容器实现，主要用于获取过滤器中的配置信息，其方法声明及说明如表 7.6 所示。

表 7.6 FilterConfig 接口的方法声明及说明

方法声明	说明
public String getFilterName()	用于获取过滤器名
public ServletContext getServletContext()	获取 Servlet 上下文
public String getInitParameter(String name)	获取过滤器的初始化参数值
public Enumeration getInitParameterNames()	获取过滤器的所有初始化参数

（3）FilterChain 接口

FilterChain 接口由 Servlet 容器实现，在这个接口中只有一个方法，其方法声明如下：

```
public void doFilter ( ServletRequest request, ServletResponse response ) throws
IOException, ServletException
```

此方法用于将过滤后的请求传递给下一个过滤器，如果此过滤器是过滤器链中的最后一个过滤器，那么请求将传送给目标资源。

7.4.3 创建并配置过滤器

创建一个过滤器对象需要实现 javax.servlet.Filter 接口及其 3 个方法，如创建名为 MyFilter 的过滤器对象的代码如下：

```
import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
/**
 * 过滤器
 */
public class MyFilter implements Filter {
    // 初始化方法
    public void init(FilterConfig fConfig) throws ServletException {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        // 初始化处理
    }
    // 过滤处理方法
    public void doFilter(ServletRequest request,
    ServletResponse response, FilterChain chain)
            throws IOException, ServletException {
            // 过滤处理
            chain.doFilter(request, response);          // 将请求向下传递
    }
    // 销毁方法
    public void destroy() {
            // 释放资源
    }
}
```

过滤器中的 init() 方法用于初始化过滤器；destroy() 方法是过滤器的销毁方法，主要用于释放资源；过滤处理的业务逻辑需要编写在 doFilter() 方法中，在请求过滤处理后需要调用 chain 参数的 doFilter() 方法将请求向下传递给下一个过滤器或目标资源。

说明　使用过滤器并不一定要将请求向下传递到下一过滤器或目标资源，如果业务逻辑需要，也可以在处理后直接回应客户端。

过滤器与 Servlet 十分相似，在创建之后同样需要对其进行配置。在 Servlet 3.0 中，提供了采用注解的方式配置过滤器，其具体的实现方法比较简单，只需要在创建类的代码上方，采用以下语法格式进行配置就可以。

```
import javax.servlet.Filter;
import javax.servlet.annotation.WebFilter;
import javax.servlet.annotation.WebInitParam;
@WebFilter(filterName = "DemoFilter",
urlPatterns = {"/*"},
initParams = {
@WebInitParam(name = "mood", value = "awake")})
public class DemoFilter implements Filter {
...
```

- filterName 属性：用于指定 Servlet 过滤器名。
- urlPatterns 属性：用于指定哪些 URL 应用该过滤器。如果指定所有页面均应用该过滤器可以设置为“/*”。
- initParams 属性：用于指定初始化参数。

说明　上面的 3 个属性不是必须的，可以根据需要选择使用。

例如，采用注解的方式配置一个作用于 index.jsp 文件的过滤器 FirstFilter，代码如下：

```
@WebFilter(filterName = "FirstFilter",urlPatterns={"/index.jsp"})
public class FirstFilter implements Filter {
```

【例 7.8】　创建一个过滤器，实现网站访问计数器的功能，并在配置过滤器时，将网站访问量的初始值设置为 5000。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

实例位置：光盘 \MR\ 源码 \ 第 7 章 \7.8

Step 1) 创建名称为 CountFilter 的类，此类实现 javax.servlet.Filter 接口，是一个过滤器对象，通过此过滤器实现统计网站访问人数功能，其关键代码如下：

```
/**
 * Servlet 过滤器实现类 CountFilter
 */
public class CountFilter implements Filter {
        private int count;                                          // 来访数量
    /**
     * 默认构造方法
     */
    public CountFilter() { }
    /**
     * 销毁方法
     */
    public void destroy() { }
    /**
     * 过滤处理方法
     */
    public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain) throws IOException, ServletException {
            count ++;                                               // 访问数量自增
            // 将 ServletRequest 转换成 HttpServletRequest
            HttpServletRequest req = (HttpServletRequest) request;
            // 获取 ServletContext
            ServletContext context = req.getServletContext();
            context.setAttribute("count", count);
                                // 将来访数量值放入到 ServletContext 中
            chain.doFilter(request, response);                      // 向下传递过滤器
    }
    /**
     * 初始化方法
     */
    public void init(FilterConfig fConfig) throws ServletException {
            String param = fConfig.getInitParameter("count");       // 获取初始化参数
            count = Integer.valueOf(param);                         // 将字符串转换为 int
    }

}
```

在 CountFilter 类中，包含了一个成员变量 count，用于记录网站访问人数，此变量在过滤器的初始化方法 init() 中被赋值，它的初始化值通过 FilterConfig 对象读取初始化参数来获取。

计数器 count 变量的值在 CountFilter 类的 doFilter() 方法被递增，因为客户端在请求服务器中的 Web 应用时，过滤器拦截请求通过 doFilter() 方法进行过滤处理，所以，当客户端请求 Web 应用时，计数器 count 的值将自增 1。为了能够访问记录器中的值，实例中将其放置于 Servlet 上下文之中，Servlet 上下文对象通过将 ServletRequest 转换成为 HttpServletRequest 对象后获取。

说明 编写过滤器对象需要实现 javax.servlet.Filter 接口，实现此接口后需要对 Filter 对象的三个方法进行实现，在这三个方法中，除了 doFilter() 方法外，如果在业务逻辑中不涉及初始化方法 init() 与销毁方法 destroy()，可以不编写任何代码对其进行实现，如实例中的 destroy() 方法。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Step 2) 配置已创建的 CountFilter 对象，此操作通过注解来实现，其关键代码如下：

```
@WebFilter(
          urlPatterns = { "/index.jsp" },
          initParams = {
                    @WebInitParam(name = "count", value = "5000")
          })
public class CountFilter implements Filter {
```

CountFilter 对象的配置主要通过声明过滤器及创建过滤器的映射进行实现，其中声明过滤器通过 <filter> 标签进行实现，在声明过程中，实例通过 initParams 属性配置过滤器的初始化参数，初始化参数的名称为 count，参数值为 5000。

说明 如果直接对过滤器对象中的成员变量进行赋值，那么在过滤器被编译后将不可修改，所以，实例中将过滤器对象中的成员变量定义为过滤器的初始化参数，从而提高代码的灵活性。

Step 3) 创建程序中的首页 index.jsp 页面，在此页面中通过 JSP 内置对象 Application 获取计数器的值，其关键代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
… <!-- 此处省略了部分 HTML 代码 -->
    <h2>
    欢迎光临，<br>
    您是本站的第【
    <%=application.getAttribute("count") %>
    】位访客！
    </h2>
</body>
</html>
```

由于在配置过滤器时，将访客人数的初始值设置为 5000，所以实例运行后，计数器的数值将在 5000 的基础上进行累加，在多次刷新页面后，实例运行结果如图 7.17 所示。



图 7.17　实现网站计数器

7.4.4　字符编码过滤器

在 Java Web 程序开发中，由于 Web 容器内部所使用编码格式不支持中文字符集，所以处理浏览器请求中的中文数据就会出现乱码现象，请求中的编码如图 7.18 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

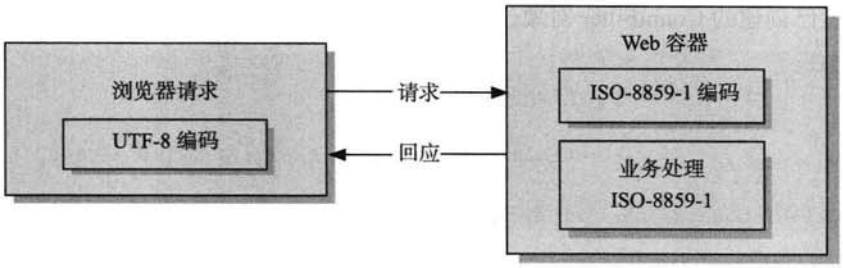


图 7.18　Web 请求中的编码

从图中可以看出，由于 Web 容器使用了 ISO-8859-1 的编码格式，所以在 Web 应用的业务处理中也会使用该编码格式。虽然浏览器提交的请求使用的是中文编码格式 UTF-8，但经过业务处理中的 ISO-8859-1 编码，仍然会出现中文乱码现象。在业务处理中重新指定中文字符集编码，即可解决这个问题。在实际开发过程中，如果通过在每一个业务处理中指定中文字符集编码，操作过于烦琐，而且容易遗漏某一个业务中的字符编码设置。如果通过过滤器来处理字符编码，即可做到简单又万无一失，如图 7.19 所示。

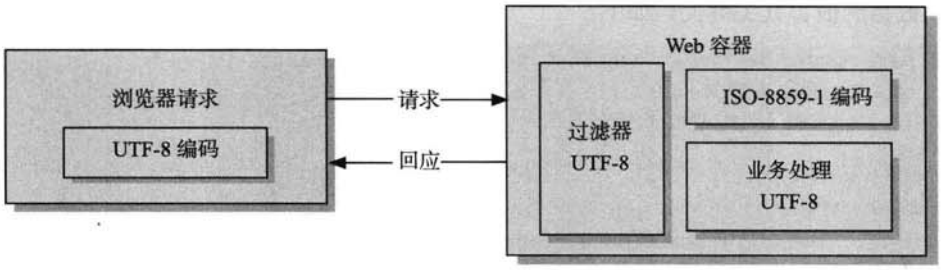


图 7.19　在 Web 容器中添加字符编码过滤器

在 Web 应用中部署了字符编码过滤器以后，即使 Web 容器的编码格式不支持中文，但浏览器的每一次请求都会经过过滤器转码，所以可完全避免中文乱码现象的产生。

【例 7.9】 实现添加图书信息功能并创建字符编码过滤器，避免产生中文乱码现象。

实例位置：光盘 \MR\ 源码 \ 第 7 章 \7.9

Step 1) 创建字符编码过滤器对象，其名称为 CharactorFilter 类。此类继承 javax.servlet.Filter 接口，并在 doFilter() 方法中对请求中的字符编码格式进行设置，其关键代码如下：

```
/**
 * Servlet 过滤器实现类 CharactorFilter
 */
public class CharactorFilter implements Filter {
   String encoding = null;                                    // 字符编码
    public CharactorFilter() {
    }
    /**
     * 销毁方法
     */
    public void destroy() {
           encoding = null;
    }
    /**
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
     * 过滤处理方法
     */
    public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain) throws IOException, ServletException {
            if(encoding != null){                                    // 判断字符编码是否为空
                request.setCharacterEncoding(encoding);  // 设置请求的编码格式
                // 设置 response 字符编码
                    response.setContentType("text/html;
                    charset="+encoding);
            }
            chain.doFilter(request, response);                       // 传递给下一个过滤器
    }
    /**
     * 初始化方法
     */
    public void init(FilterConfig fConfig) throws ServletException {
            encoding = fConfig.getInitParameter("encoding");  // 获取初始化参数
    }
}
```

CharactorFilter 类是实例中的字符编码过滤器，它主要通过在 doFilter() 方法中，指定 request 与 reponse 两个参数的字符集 encoding 进行编码处理，使得目标资源的字符集支持中文。其中 encoding 是 CharactorFilter 类定义的字符编码格式成员变量，此变量在过滤器的初始化方法 init() 中被赋值，它的值是通过 FilterConfig 对象读取配置文件中的初始化参数获取的。

注意　在过滤器对象的 doFilter() 方法中，业务逻辑处理完成之后，需要通过 FilterChain 对象的 doFilter() 方法将请求传递到下一过滤器或目标资源，否则将出现错误。

在创建了过滤器对象之后，还需要对过滤器进行一定的配置才可以正常使用，采用注解的方式配置过滤器 CharactorFilter 的代码如下：

```
@WebFilter(
            urlPatterns = { "/*" },
            initParams = {
                    @WebInitParam(name = "encoding", value = "UTF-8")
            })                                                       // 配置过滤器
```

在过滤器 CharactorFilter 的配置声明中，将它的初始化参数 encoding 的值设置为 UTF-8，它与 JSP 页面的编码格式相同，支持中文。

说明　配置过滤器时，URL 映射可以使用正则表达式进行配置，如本实例中使用“/*”来匹配所有请求。

Step 2) 创建名称为 AddServlet 的类，该类继承 HttpServlet，是处理添加图书信息请求的 Servlet 对象，其关键代码如下：

```
/**
 * Servlet implementation class AddServlet
 */
@WebServlet("/AddServlet")        // 配置 Servlet
public class AddServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public AddServlet() {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        super();
    }
    /**
     * 处理 GET 请求的方法
     */
    protected void doGet(HttpServletRequest request,
                HttpServletResponse response) throws ServletException,
                IOException {
        doPost(request, response);                              // 处理 GET 请求
    }
    /**
     * 处理 POST 请求的方法
     */
    protected void doPost(HttpServletRequest request,
                HttpServletResponse response) throws ServletException,
                IOException {
        // 处理 POST 请求
        PrintWriter out = response.getWriter();                 // 获取 PrintWriter
        String id = request.getParameter("id");                 // 获取图书编号
        String name = request.getParameter("name");             // 获取名称
        String author = request.getParameter("author");         // 获取作者
        String price = request.getParameter("price");           // 获取价格
        out.print("<h2> 图书信息添加成功 </h2><hr>");             // 输出图书信息
        out.print(" 图书编号: " + id + "<br>");
        out.print(" 图书名称: " + name + "<br>");
        out.print(" 作者: " + author + "<br>");
        out.print(" 价格: " + price + "<br>");
        out.flush();                                            // 刷新流
        out.close();                                            // 关闭流
    }
}
```

AddServlet 类主要通过 doPost() 方法实现添加图书信息请求的处理，其处理方式是将所获取到的图书信息数据直接输出到页面中。

说明 通常情况下，Servlet 所处理的请求类型都是 GET 或 POST，所以可以在 doGet() 方法中调用 doPost() 方法，把业务处理代码写到 doPost() 方法中，或在 doPost() 方法中调用 doGet() 方法，把业务处理代码写到 doGet() 方法中，无论 Servlet 接收的请求类型是 GET 还是 POST，Servlet 都对其进行处理。

Step 3) 创建名称为 index.jsp 的文件，它是程序中的首页，主要用于放置添加图书信息的表单。index.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
… <!-- 此处省略了部分 HTML 和 CSS 代码 -->
    <section>
        <h2>     添加图书信息 </h2>
        <form action="AddServlet" method="post">
            <ul>
                <li> 图书编号: <input type="text" name="id"></li>
                <li> 图书名称: <input type="text" name="name"></li>
                <li> 作    者: <input type="text" name="author"></li>
                <li> 价    格: <input type="text" name="price"></li>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                        <li>        <input type="submit" value="添  加"></li>
                    </ul>
                </form>
            </section>
        </body>
        </html>
```

运行本实例，将打开 index.jsp 页面，输入如图 7.20 所示的图书信息后，单击“添加”按钮，将显示如图 7.21 所示的效果。



图 7.20　添加图书信息



图 7.21　显示图书信息

虽然在 AddServlet 中没有指定 request 与 response 的字符编码格式，但在图中没有看到中文乱码现象，原因是在程序中使用字符编码过滤器指定了编码格式。

7.5　Servlet监听器

在 Servlet 技术中已经定义了一些事件，并且可以针对这些事件来编写相关的事件监听器，从而使用监听器处理事件。

7.5.1　Servlet 监听器简介

监听器的作用是监听 Web 容器的有效期事件，由容器管理。利用 Listener 接口监听在容器中的某个执行程序，并且根据其应用程序的需求做出适当响应。表 7.7 所示为 Servlet 和 JSP 中的 8 个 Listener 接口和 6 个 Event 类。

表 7.7　Listener 接口与 Event 类

Listener 接口	Event 类
ServletContextListener	ServletContextEvent
ServletContextAttributeListener	ServletContextAttributeEvent
HttpSessionListener	HttpSessionEvent
HttpSessionActivationListener	
HttpSessionAttributeListener	HttpSessionBindingEvent
HttpSessionBindingListener	

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（续）

Listener 接口	Event 类
ServletRequestListener	ServletRequestEvent
ServletRequestAttributeListener	ServletRequestAttributeEvent

7.5.2 Servlet 监听器的工作原理

Servlet 监听器是当今 Web 应用开发的一个重要组成部分，它在 Servlet 2.3 规范中与 Servlet 过滤器一起引入。并且在 Servlet 3.0 规范中进行了较大改进，主要用来监听和控制 Web 应用，极大地增强了 Web 应用的事件处理能力。

Servlet 监听器的功能比较接近 Java 的 GUI 程序的监听器，可以监听 Servlet 容器由于 Web 应用中状态改变而产生的相应事件，然后接受并处理这些事件。

7.5.3 监听 Servlet 上下文

Servlet 上下文监听可以监听 ServletContext 对象的创建、删除和添加属性，以及删除和修改操作，该监听器需要用到如下两个接口。

1. ServletContextListener 接口

该接口存放在 javax.servlet 包内，主要监听 ServletContext 的创建和删除。它提供了如下两个方法，也称为“Web 应用程序的生命周期方法”。

1）contextInitialized（ServletContextEvent event）方法：通知正在收听的对象应用程序已经被加载及初始化。

2）contextDestroyed（ServletContextEvent event）方法：通知正在收听的对象应用程序已经被载出，即关闭。

2. ServletAttributeListener 接口

该接口存放在 javax.servlet 包内，主要监听 ServletContext 属性的增加、删除及修改，它提供了如下 3 个方法。

1）attributeAdded（ServletContextAttributeEvent event）方法：若有对象加入 Application 的范围，通知正在收听的对象。

2）attributeReplaced（ServletContextAttributeEvent event）方法：若在 Application 范围内的一个对象取代另一个对象，通知正在收听的对象。

3）attributeRemoved（ServletContextAttributeEvent event）方法：若有对象从 Application 的范围移除，通知正在收听的对象。

7.5.4 监听 HTTP 会话

有如下 4 个接口可以监听 HTTP 会话（HttpSession）信息。

1. HttpSessionListener 接口

该接口监听 HTTP 会话的创建及销毁，它提供了如下两个方法。

1）sessionCreated（HttpSessionEvent event）方法：通知正在收听的对象，session 已经被加载及初始化。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2）sessionDestroyed（HttpSessionEvent event）方法：通知正在收听的对象，session 已经被载出（HttpSessionEvent类的主要方法是getSession()，可以使用该方法回传一个session对象）。

2．HttpSessionActivationListener 接口

该接口实现监听 HTTP 会话 active 和 passivate 情况，它提供了如下 3 个方法。

1）attributeAdded（HttpSessionBindingEvent event）方法：若有对象加入 session 的范围，通知正在收听的对象。

2）attributeReplaced（HttpSessionBindingEvent event）方法：若在 session 范围内的一个对象取代另一个对象，通知正在收听的对象。

3）attributeRemoved（HttpSessionBindingEvent event）方法：若有对象从 session 的范围移除，通知正在收听的对象（HttpSessionBindingEvent 类主要有 3 个方法，即 getName()、getSession() 和 getValues()）。

3．HttpBindingListener 接口

该接口实现监听 HTTP 会话中对象的绑定信息，它是唯一不需要在 web.xml 中设置 Listener 的，并提供了以下两个方法。

1）valueBound（HttpSessionBindingEvent event）方法：当有对象加入 session 的范围时，会被自动调用。

2）valueUnBound（HttpSessionBindingEvent event）方法：当有对象从 session 的范围内移除时，会被自动取消调用。

4．HttpSessionAttributeListener 接口

该接口实现监听 HTTP 会话中属性的设置请求，它提供了如下两个方法。

1）sessionDidActivate（HttpSessionEvent event）方法：通知正在收听的对象，其 session 已经变为有效状态。

2）sessionWillPassivate（HttpSessionEvent event）方法：通知正在收听的对象，其 session 已经变为无效状态。

7.5.5　监听 Servlet 请求

在 Servlet 2.4 规范中新增加了一种技术，即监听客户端的请求。一旦能够在监听程序中获取客户端的请求，即可统一处理请求。要监听客户端的请求和请求参数，需要实现如下两个接口。

1．ServletRequestListener 接口

该接口提供了如下两个方法。

1）requestInitalized（ServletRequestEvent event）方法：通知正在收听的对象，ServletRequest 已经被加载及初始化。

2）requestDestroyed（ServletRequestEvent event）方法：通知正在收听的对象，ServletRequest 已经被载出，即关闭。

2．ServletRequestAttributeListener 接口

该接口提供了如下 3 个方法。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

1）attributeAdded（ServletRequestAttributeEvent event）方法：若有对象加入 request 的范围，通知正在收听的对象。

2）attributeReplaced（ServletRequestAttributeEvent event）方法：若在 request 范围内的一个对象取代另一个对象，通知正在收听的对象。

3）attributeRemoved（ServletRequestAttributeEvent event）方法：若有对象从 request 的范围移除，通知正在收听的对象。

7.5.6 使用监听器查看在线用户

利用 Listener 接口可以监听某个执行程序，并根据该程序的需求做出适当响应。

【例 7.10】 通过监听器查看用户在线情况。

实例位置：光盘 \MR\ 源码 \ 第 7 章 \7.10

Step 1) 创建 UserInfoList.java 类文件，用来保存在线用户和对其执行具体操作，该文件的完整代码如下：

```
public class UserInfoList {
    private static UserInfoList user = new UserInfoList();
    private Vector vector = null;
    /*
         利用 private 调用构造函数，防止外界产生新的 instance 对象
     */
    public UserInfoList() {
        this.vector = new Vector();
    }
    /* 外界使用的 instance 对象 */
    public static UserInfoList getInstance() {
        return user;
    }
    /* 增加用户 */
    public boolean addUserInfo(String user) {
        if (user != null) {
          this.vector.add(user);
          return True;
        } else {
            return False;
        }
    }
    /* 获取用户列表 */
    public Vector getList() {
        return vector;
    }
    /* 移除用户 */
    public void removeUserInfo(String user) {
        if (user != null) {
          vector.removeElement(user);
        }
    }
}
```

Step 2) 创建 UserInfoTrace.java 类文件，主要实现 valueBound(HttpSessionBindingEvent arg0) 和 valueUnbound(HttpSessionBindingEvent arg0) 两个方法。当有对象加入 session

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

时，会自动执行 valueBound() 方法；当有对象从 session 中移除时，会自动执行 valueUnbound() 方法。在 valueBound() 和 valueUnbound() 方法中都加入了输出信息的功能，使用户在控制台中更清楚地了解执行过程，该文件的完整代码如下：

```
public class UserInfoTrace implements javax.servlet.http.
      HttpSessionBindingListener {
    private String user;
    private UserInfoList container = UserInfoList.getInstance();
    public UserInfoTrace() {
      user = "";
    }
    /* 设置在线监听人员 */
    public void setUser(String user) {
      this.user = user;
    }
    /* 获取在线监听 */
    public String getUser() {
      return this.user;
    }
    public void valueBound(HttpSessionBindingEvent arg0) {
      System.out.println("上线 " + this.user);
    }
    public void valueUnbound(HttpSessionBindingEvent arg0) {
      System.out.println("下线 " + this.user);
      if (user != "") {
        container.removeUserInfo(user);
        }
    }
}
```

Step 3) 创建 showUser.jsp 页面文件，在其中设置 session 的 setMaxInactiveInterval() 为 10 秒。这样可以缩短 session 的生命周期，该页面文件的关键代码如下：

```
<%@ page import="java.util.*"%>
<%@ page import="com.listener.*"%>
<%
UserInfoList list = UserInfoList.getInstance();  // 获得 UserInfoList 类的对象
UserInfoTrace ut = new UserInfoTrace();           // 创建 UserInfoTrace 类的对象
request.setCharacterEncoding("UTF-8");            // 设置编码为 UTF-8，解决中文乱码
String name = request.getParameter("user");       // 获取输入的用户名
ut.setUser(name);                                 // 设置用户名
session.setAttribute("list", ut);                // 将 UserInfoTrace 对象绑定到 Session 中
list.addUserInfo(ut.getUser());                   // 添加用户到 UserInfo 类的对象中
session.setMaxInactiveInterval(30);               // 设置 Session 的过期时间为 30 秒
%>
<textarea rows="8" cols="20">
<%
Vector vector=list.getList();
if(vector!=null&&vector.size()>0){
for(int i=0;i<vector.size();i++){
  out.println(vector.elementAt(i));
}
}
%>
</textarea>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

运行本实例，在用户登录页面中输入用户名，如图 7.22 所示，单击“登录”按钮，将进入到如图 7.23 所示的在线用户列表页面，在该页面中，将显示当前在线用户。同时，在控制台中将输出如图 7.24 所示的信息。



图 7.22　用户登录页面



图 7.23　在线用户列表

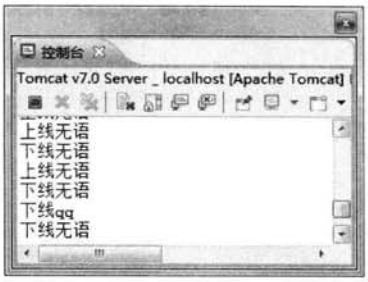


图 7.24　控制台输出信息

7.6　上机实践

7.6.1　记录用户访问次数

➤➤➤ 题目描述

在浏览网站时有些网站会有计数器的功能，浏览者每访问一次网站，计数器就累加一次。本上机实践尝试编写一个记录用户访问次数的网站，运行结果如图 7.25 所示。

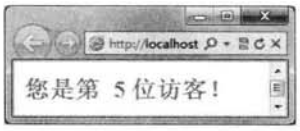


图 7.25　运行结果

➤➤➤ 技术指导

实现计数器主要是在 Servlet 中应用 ServletContext 接口，当 Servlet 容器在启动一个 Web 应用时会为其创建一个 ServletContext 对象；当 Servlet 容器终止一个 Web 应用时，该对象也会被销毁。所以它与 Web 应用程序有同样的生命周期，即整个 Web 应用组件可以共享 ServletContext 对象中存放的共享数据。在 ServletContext 接口中存取共享数据的方法如下。

1）setAttribute(String name,Object object)：在 ServletContext 对象中存放共享数据，参数 name 为属性名，object 为属性值。

2）removeAttribute(String name)：根据指定的参数 name 属性名删除 ServletContext 对象中的共享数据。

3）getAttribute(String name)：根据指定的参数 name 属性获取 ServletContext 对象中的共享数据。

7.6.2　防盗链过滤器

➤➤➤ 题目描述

在实际的开发中，为了防止本网站内的资源被其他网站盗链下载，可以编写一个防盗链过滤器。本上机实践要求尝试编写一个防盗链过滤器，防止通过其他 URL 地址直接访问本站资源。运行效果如图 7.26 和图 7.27 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 7.26　本站访问时的运行结果



图 7.27　盗链网站内的图片时的运行结果

说明　程序编写完毕并运行后，可以看到图 7.26 所示的运行结果，如果编写一个本地的 HTML 页，在该文件中，通过 访问实例中的图片，就可以看到如图 7.27 所示的运行结果。

➤➤➤ 技术指导

在实现本实例时，可以应用 request 对象的 getHeader() 方法获取信息头来源地址，若地址来自其他网站就弹出错误图片。getHeader() 方法的语法结构如下：

```
public String getHeader(String headerName)
```

- headerName：指定字符串类型的响应头名称。在实现防盗链功能时，将其指定为 Referer。需要注意的是，Referer 的首字母要大写，否则在 Windows 7 系统下，只有通过 localhost 访问时才能获取到值。

7.6.3　敏感词过滤器

➤➤➤ 题目描述

在开发论坛等网站时，需要对输入的内容进行控制，防止输出非法内容或者敏感内容，常规的办法是保存数据库之前对非法内容进行替换，但这种办法具有局限性，工作量大并且耦合性比较高。为了解决这一问题，可以编写一个过滤器，对非法字符和关键字进行过滤处理。运行结果如图 7.28 所示。

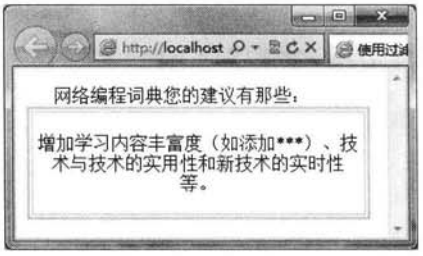


图 7.28　敏感词过滤器

➤➤➤ 技术指导

要实现本实例，可以在服务器端使用过滤器技术。在 Web 服务器获得用户的请求之前，过滤器可以访问该请求。在 Web 服务器将输出响应发送给用户之前，过滤器还可以访问该响应。所以在过滤器中访问该响应信息，然后将该响应转换为自定义的响应，最后将过滤后的自定义响应内容返回给客户端。

在 java.servlet.http 包中，包含了一个名为 HttpServletResponseWrapper 的类，该类的对象表示一个自定义的响应对象，它实现了 HttpServletResponse 接口，其构造方法通过传入的 HttpServletResponse 类型的参数，将响应转换为自定义的响应。构造方法的语法结构如下：

```
public HttpServletResponseWrapper(HttpServletResponse response)
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

7.7 小结

本章主要介绍了 Servlet 原理、Servlet 过滤器原理及二者的实际应用。在 Java EE 中，Servlet 是 Java EE API 中的核心，更是 Web 开发中不可脱离的组成部分。学习本章内容读者必须掌握 Servlet、过滤器的基本原理和字符编码过滤器。

7.8 习题

选择题

1. HttpServlet 的子类要从 HTTP 请求中获得请求参数，应该调用哪个方法？（　　）
 A. 调用 HttpServletRequest 对象的 getAttribute() 方法
 B. 调用 ServletContext 对象的 getAttribute() 方法
 C. 调用 HttpServletRequest 对象的 getParameters() 方法
 D. 调用 HttpServletRequest 对象的 getHeader() 方法
2. Servlet 容器在启动 Web 应用时创建哪些对象（多选）？（　　）
 A. ServletRequest 对象　　B. ServletContext 对象
 C. Filter 对象　　D. 所有的 Servlet 对象
3. 以下哪些对象由 Servlet 容器创建（多选）？（　　）
 A. Servlet 对象　　B. ServletRequest 对象
 C. ServletConfig 对象　　D. ServletResponse 对象
4. 当响应正文为哪种数据类型时，浏览器会让用户下载响应正文？（　　）
 A. text/html　　B. text/plain
 C. application/msword　　D. image/jpeg
5. 当 Servlet 容器销毁一个 Servlet 时，会销毁哪些对象（多选）？（　　）
 A. Servlet 对象　　B. 与 Servlet 对象关联的 ServletConfig 对象
 C. ServletContext 对象　　D. ServletRequest 对象和 ServletResponse 对象

填空题

1. Servlet 的生命周期的三个阶段分别为_____阶段、_____阶段和_____阶段。
2. 假设 Form 表单的提交方式为 GET，那么在 Servlet 中调用的是_____方法。
3. Servlet 在调用 doGet() 或 doPost() 方法之前，首先调用_____方法。
4. 在 Servlet 中，希望响应正文的类型为图片，应该设置 response.setContentType() 的方法参数为_____。
5. Filter 接口的 doFilter() 方法的参数类型包括_____。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 8 章　使用表达式语言（EL）

视频讲解：72 分钟

表达式语言的英文为 Expression Language，简称 EL，它提供了一个让 Web 页和 Bean 管理进行通信的重要机制，被应用在 JSF 和 JSP 技术中。通过它可以简化 JSP 开发中对象的引用，从而规范页面代码，增加程序的可读性及可维护性。本章将对 EL 表达式语言的语法、运算符及隐含对象进行详细介绍。

通过学习本章，读者可以达到以下学习目的：

- 了解 EL 的基本语法；
- 了解禁用 EL 的几种方法；
- 了解 EL 表达式中的保留关键字；
- 掌握 EL 的运算符及优先级；
- 掌握 EL 的隐含对象的使用；
- 掌握如何定义和使用 EL 的函数。

8.1　EL概述

在 EL 出现前，开发 Java Web 应用时经常需要将大量 Java 代码片段嵌入到 JSP 页面中。这会使页面看起来很乱，而使用 EL 则比较简洁。例如，我们需要在 JSP 页面中显示保存在 session 范围内的变量 username。如果使用 Java 代码片段，则代码如下：

```
<%if(session.getAttribute("username")!=null){
    out.println(session.getAttribute("username").toString());
} %>
```

而使用 EL，则只需要下面的一行代码即可实现：

```
${username }
```

因此 EL 在 Web 开发中比较常用，它通常与 JSTL 一同使用。EL 表达式提供了获取对象及属性的简单方式，还可以支持简单的运算，包括比较运算等。

8.1.1　EL 的基本语法

EL 的基本语法很简单，它以“${”开头，以“}”结束，中间为合法的表达式，其语法格式如下：

```
${expression}
```

expression 指定要输出的内容，可以是字符串或 EL 运算符组成的表达式。

技巧　由于 EL 表达式的语法以“${”开头，所以如果在 JSP 网页中要显示“${”字符串，必须在前面加上 \ 符号。即 \${，或者写成 ${'${'}，就是用表达式来输出“${”符号串。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在 EL 中要输出一个字符串，可以将其放在一对单引号或双引号内。例如，要在页面中输出字符串“明日科技编程词典”，可以使用下面任意一行代码：

```
${'明日科技编程词典'}
${"明日科技编程词典"}
```

8.1.2 EL 的特点

EL 除了具有语法简单和使用方便的特点之外，还具有以下特点。

- 可以与 JSTL 及 JavaScript 语句结合使用。
- 自动执行类型转换。如果想通过 EL 输入两个字符串型数值（如 number1 和 number2）的和，可以通过“+”连接（如 ${number1+number2}）。
- 不仅可以访问一般变量，而且还可以访问 JavaBean 中的属性、嵌套属性和集合对象。
- 可以执行算术运算、逻辑运算、关系运算和条件运算等。
- 可以获得命名空间（PageContext 对象，它是页面中所有其他内置对象的最大范围的集成对象，通过它可以访问其他内置对象）。
- 在执行除法运算时，如果 0 作为除数，则返回无穷大 Infinity，而不返回错误。
- 可以访问 JSP 的作用域（request、session、application 及 page）。
- 扩展函数可以与 Java 类的静态方法执行映射。

8.2 与低版本的环境兼容——禁用EL

如今 EL 已经是一项成熟且标准的技术，只要安装的 Web 服务器能够支持 Servlet 2.4/JSP 2.0，即可在 JSP 页面中直接使用 EL。由于在 JSP 2.0 以前版本中没有 EL，为了和以前的规范兼容，在 JSP 中提供了 3 种禁用 EL 的方法。

技巧 如果在使用 EL 时没有正确解析其内容，而是原样显示在页面中（包括 $ 和 {}），则说明 Web 服务器不支持 EL，此时需要检查 EL 有没有被禁用。

8.2.1 使用斜杠符号“\”

该方法只需在 EL 的开始标记“${”前加上“\”符号，其语法格式如下：

```
\${expression}
```

例如，要禁用页面中的 EL“${number}”，可以使用下面的代码：

```
\${number}
```

说明 该方法仅适合禁用页面中一个或多个 EL 表达式的情况。

8.2.2 使用 page 指令

使用 JSP 的 page 指令也可以禁用 EL，其语法格式如下：

```
<%@ page isELIgnored="布尔值" %>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

isELIgnored 属性指定是否禁用页面中的 EL，如果为 true，则忽略页面中的 EL，否则将解析页面中的 EL。

例如，如果要忽略页面中的 EL，可以在页面的顶部添加以下代码：

```
<%@ page isELIgnored="true" %>
```

说明　该方法适合禁用一个 JSP 页面中的 EL。

8.2.3　在 web.xml 文件中配置 <el-ignored> 元素

在 web.xml 文件中配置 <el-ignored> 元素可以禁用服务器中的 EL，其代码如下：

```
<jsp-config>
   <jsp-property-group>
          <url-pattern>*.jsp</url-pattern>
          <el-ignored>true</el-ignored><!-- 将此处的值设置为 false，表示使用 EL-->
   </jsp-property-group>
</jsp-config>
```

说明　该方法适用于禁用 Web 应用中所有 JSP 页面中的 EL。

8.3　保留关键字

EL 有保留关键字，在为变量命名时应该避免使用这些关键字，包括使用 EL 输出已经保存在作用域范围内的变量。如果已经定义，那么需要将其修改为其他变量名。EL 的保留关键字如表 8.1 所示。

表 8.1　EL 的保留关键字

保留关键字			
and	eq	gt	true
instanceof	div	or	ne
le	false	empty	mod
not	lt	ge	null

如果在 EL 中使用了保留关键字，那么在 Eclipse 中显示如图 8.1 所示的错误信息。

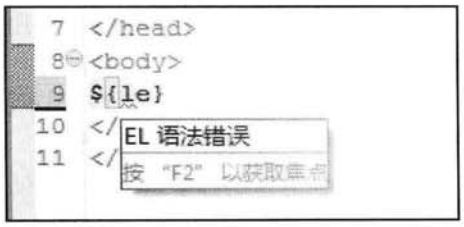


图 8.1　在 Eclipse 中显示的错误信息

如果忽略该提示直接运行程序，显示如图 8.2 所示的错误信息。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 8.2　在 IE 浏览器中显示的错误信息

8.4　EL运算符及其优先级

EL 提供了访问数据运算符、算术运算符、关系运算符、逻辑运算符、条件运算符及 empty 运算符等，其优先级如图 8.3 所示。

运算符的优先级决定了在多个运算符同时存在时各个运算符的求值顺序，同级运算符采用从左向右计算的原则。

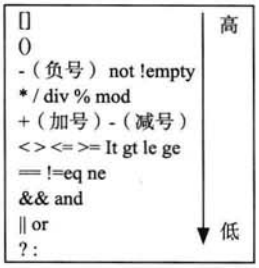


图 8.3　EL 运算符的优先级

说明　使用括号“()”可以改变优先级，如 ${5/(9-6)} 改变了先乘除且后加减的基本规则，这是因为括号内的优先级高于绝大部分的运算符。在复杂的表达式中使用括号可以使表达式更容易阅读且避免出错。

8.4.1　通过 EL 访问数据

通过 EL 提供的 [] 和 . 运算符可以访问数据，通常情况下二者是等价的，可以相互代替。例如，要访问 JavaBean 对象 userInfo 的 id 属性，可以写成以下两种形式：

```
${userInfo.id}
${userInfo[id]}
```

但在有些情况下这两种运算符不可相互替代，如当对象的属性名中包括一些特殊符号（- 或 .）时，只能使用 [] 运算符来访问对象的属性。例如，${userInfo[user-id]} 是正确的，而 ${userInfo.user-name} 则是错误的。另外，EL 的 [] 运算符还可以用来读取数组或是 List 集合中的数据。

应用 [] 运算符可以获取数组的指定元素，但是 . 运算符则不能。例如，要获取 request 范围中的数组 arrBook 中的第 1 个元素，可以使用以下 EL 表达式：

```
${arrBook[0]}
```

说明　由于数组的索引值从 0 开始，所以要获取第 1 个元素，需要使用索引值 0。

【例 8.1】　通过 EL 输出数组的全部元素。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

实例位置：光盘 \MR\ 源码 \ 第 8 章 \8.1

编写 index.jsp 文件，在其中首先定义一个包含 3 个元素的一维数组并赋初始值，然后通过 for 循环和 EL 输出该数组中的全部元素。该文件的关键代码如下 .

```
<%
String[] arr={"Java Web 开发典型模块大全 ","Java 范例完全自学手册 ","JSP 项目开发全程实录 "};
                                                        // 定义一维数组
request.setAttribute("book",arr);                       // 将数组保存到 request 对象中
String[] arr1=(String[])request.getAttribute("book");
                                        // 获取保存到 request 范围内的变量
// 通过循环和 EL 输出一维数组的内容
for(int i=0;i<arr1.length;i++){
   request.setAttribute("requestI",i);   // 将循环变量 i 保存到 request 范围内的变量中
%>
   ${requestI}: ${book[requestI]}<br>                   <!-- 输出数组中第 i 个元素 -->
<%} %>
```

说明　在上面的代码中必须将循环变量 i 保存在 request 范围内的变量中，否则将不能正确访问数组。这里不能直接使用 Java 代码片段中定义的变量 i，也不能使用 <%=i%> 输出 i。

在运行时，系统会先获取 requestI 变量的值，然后将输出数组内容的表达式转换为 "${fruit[索引]}" 格式（例如，获取第 1 个数组元素，则转换为 ${fruit[0]} ）后输出。实例运行结果如图 8.4 所示。

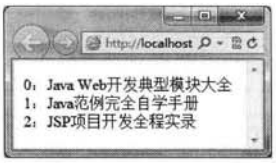


图 8.4　实例运行结果

应用 [] 运算符还可以获取 List 集合中的指定元素，但是 . 运算符则不能。

【例 8.2】 在 session 域中保存一个包含 3 个元素的 List 集合对象，并应用 EL 输出该集合的全部元素的代码如下。

实例位置：光盘 \MR\ 源码 \ 第 8 章 \8.2

```
<%
List<String> list = new ArrayList<String>();     // 声明一个 List 集合的对象
list.add(" 饼干 ");                               // 添加第 1 个元素
list.add(" 牛奶 ");                               // 添加第 2 个元素
list.add(" 果冻 ");                               // 添加第 3 个元素
session.setAttribute("goodsList",list);           // 将 List 集合保存到 session 对象中
List<String> list1=(List<String>)session.getAttribute("goodsList");
                                                  // 获取保存在 session 范围内的变量
// 通过循环和 EL 输出 List 集合的内容
for(int i=0;i<list1.size();i++){
   request.setAttribute("requestI",i);            // 将循环增量保存在 request 范围内
%>
   ${requestI}: ${goodsList[requestI]}<br>        <!-- 输出集合中的第 i 个元素 -->
<%} %>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

运行后显示如图 8.5 所示的运行结果。

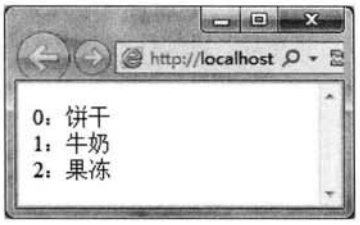


图 8.5　应用 EL 遍历 List 集合

8.4.2　在 EL 中执行算术运算

EL 提供了加、减、乘、除和求余 5 种算术运算符，如表 8.2 所示。

表 8.2　EL 的算术运算符

运 算 符	功　能	示　例	结　果
+	加	${19+1}	20
-	减	${66-30}	36
*	乘	${52.1*10}	521.0
/ 或 div	除	${5/2} 或 ${5 div 2}	2.5
		${9/0} 或 ${9 div 0}	Infinity
% 或 mod	求余	${17%3} 或 ${17 mod 3}	2
		${15%0} 或 ${15 mod 0}	将抛出异常 java.lang.ArithmeticException: / by zero

注意　EL 的+运算符不能连接两个字符串，如果用其连接两个不可以转换为数值型的字符串，将抛出异常；如果连接两个可以转换为数值型的字符串，则自动将这两个字符串转换为数值型后执行加法运算。

8.4.3　在 EL 中判断对象是否为空

在 EL 中判断对象是否为空可以使用 empty 运算符，该运算符是一个前缀（prefix）运算符。即其位于操作数前，用来确定一个对象或变量是否为 null 或空。该运算符的语法格式如下：

```
${empty expression}
```

expression 用于指定要判断的变量或对象。

例如，定义两个 request 范围内的变量 user 和 user1，并分别设置为 null 和 ""，代码如下：

```
<%request.setAttribute("user",""); %>
<%request.setAttribute("user1",null); %>
```

然后通过 empty 运算符判断 user 和 user1 是否为空，代码如下：

```
${empty user}                    <!-- 返回值为 true -->
${empty user1}                   <!-- 返回值为 true -->
```

说明　一个变量或对象为 null 或空代表的意义不同，null 表示这个变量没有指定任何对象；而空表示这个变量所属对象的内容为空，如空字符串、空的数组或者空的 List 容器。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



另外，empty 运算符也可以与 not 运算符结合使用，用于判断一个对象或变量是否为非空。例如，要判断 request 范围中的变量 user 是否为非空，可以使用以下代码：

```
<%request.setAttribute("user","");%>
${not empty user}                                    <!-- 返回值为 false -->
```

8.4.4　在 EL 中执行逻辑关系运算

在 EL 中关系运算符用于比较两个表达式，表达式可以是数值型和字符串型；逻辑运算符则常用于处理 boolean 型数据。这两个运算符经常一同使用，如可以用下面的表达式判断 60 ~ 80 分的成绩：

```
成绩 >60 and 成绩 <80
```

在这个表达式中，> 和 < 为关系运算符，and 为与运算符。

（1）关系运算符

在 EL 中提供了 6 种关系运算符，不仅可以用来比较整数和浮点数，还可以用来比较字符串，其使用格式如下：

```
${ 表达式 1 关系运算符 表达式 2}
```

这些关系运算符如表 8.3 所示。

表 8.3　EL 的关系运算符

运　算　符	功　能	示　例	结　果
== 或 eq	等于	${10==10} 或 ${10 eq 10}	true
		${"A"=="a"} 或 ${"A" eq "a"}	false
!= 或 ne	不等于	${10!=10} 或 ${10 ne 10}	false
		${"A"!="A"} 或 ${"A" ne "A"}	false
< 或 lt	小于	${7<6} 或 ${7 lt 6}	false
		${"A"<"B"} 或 ${"A" lt "B"}	true
> 或 gt	大于	${7>6} 或 ${7 gt 6}	true
		${"A">"B"} 或 ${"A" gt "B"}	false
<= 或 le	小于等于	${7<=6} 或 ${7 le 6}	false
		${"A"<="A"} 或 ${"A" le "A"}	true
>= 或 ge	大于等于	${7>=6} 或 ${7 ge 6}	true
		${"A">="B"} 或 ${"A" ge "B"}	false

（2）逻辑运算符

在执行比较运算时，如果涉及两个或两个以上的条件判断（例如，要判断变量 a 是否大于等于 60，并且小于等于 80），则需要应用逻辑运算符。其条件表达式的值必须是 boolean 型或可以转换为 boolean 型的字符串，并且返回的结果也是 boolean 型。

EL 的逻辑运算符如表 8.4 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 8.4 EL 的逻辑运算符

运 算 符	功 能	示 例	结 果
&& 或 and	与	${true && false} 或 ${true and false}	false
		${"true" && "true"} 或 ${"true" and "true"}	true
\|\| 或 or	或	${true \|\| false} 或 ${true or false}	true
		${false \|\| false} 或 ${false or false}	false
! 或 not	非	${! true} 或 ${not true}	false
		${!false} 或 ${not false}	true

说明 在执行逻辑运算时，只要表达式的值可以确定，则停止执行。例如，在表达式 A and B and C 中，如果 A 为 true 且 B 为 false，则只计算 A and B 并返回 false；在表达式 A or B or C 中，如果 A 为 true 且 B 为 false，则只计算 A or B，并返回 ture。

【例 8.3】 编写 index.jsp 文件，在其中首先定义两个 request 范围内的变量并赋初始值，然后输入这两个变量，最后将这两个变量和关系运算符、逻辑运算符组成条件表达式并输出，其关键代码如下：

实例位置：光盘 \MR\ 源码 \ 第 8 章 \8.3

```
<%
request.setAttribute("userName","mr");     // 定义 request 范围内的变量 userName
request.setAttribute("pwd","mrsoft");      // 定义 pwd 范围内的变量 pwd
%>
userName=${userName}<br>                   <!-- 输入变量 userName -->
pwd=${pwd}<br>                             <!-- 输入变量 pwd -->
\${userName!="" and (userName=="明日")}: <!-- 将 EL 原样输出 -->
${userName!="" and userName=="明日" }<br> <!-- 输出由关系和逻辑运算符组成的表达式的值-->
\${userName=="mr" and pwd=="mrsoft" }:     <!-- 将 EL 原样输出 -->
${userName=="mr" and pwd=="mrsoft" }       <!-- 输出由关系和逻辑运算符组成的表达式的值-->
```

运行本实例，显示如图 8.6 所示的运行结果。

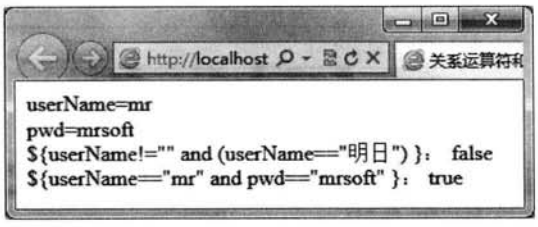


图 8.6 关系运算符和逻辑运算符的应用

8.4.5 在 EL 中执行条件运算

在 EL 中执行简单的条件运算可以通过条件运算符实现，它的优点是简单和方便，其语法格式如下：

```
${ 条件表达式 ? 表达式 1 : 表达式 2}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 条件表达式：用于指定一个条件表达式，其值为 boolean 型，可以由关系运算符、逻辑运算符和 empty 运算符组成。
- 表达式 1：用于指定当条件表达式的值为 true 时要返回的值。
- 表达式 2：用于指定当条件表达式的值为 false 时要返回的值。

说明　在上面的语法格式中，如果条件表达式为 true，则返回表达式 1 的值，否则返回表达式 2 的值。

例如，应用条件运算符实现，当变量 cart 的值为空时输出“购物车为空”，否则输出 cart 的值，代码如下：

```
${empty cart ? "cart为空" : cart}
```

说明　通常情况下条件运算符可以用 JSTL 中的条件标签 <c:if> 或 <c:choose> 替代。

8.5　EL的隐含对象

为了能够获得 Web 应用程序中的相关数据，EL 提供了 11 个隐含对象。这些对象类似 JSP 的内置对象，直接通过对象名操作。在 EL 的隐含对象中，除 PageContext 是 JavaBean 对象，对应于 javax.servlet.jsp.PageContext 类型，其他隐含对象均对应于 java.util.Map 类型。这些隐含对象可以分为页面上下文对象、访问作用域范围的隐含对象和访问环境信息的隐含对象。

8.5.1　页面上下文对象

页面上下文对象为 pageContext，用于访问 JSP 内置对象（如 request、response、out、session、exception 和 page 等，但不能用于获取 application、config 和 pageContext 对象）和 servletContext。在获取这些内置对象后，即可获取其属性值。这些属性与对象的 Getter 方法相对应，在使用时去掉方法名中的 get，并将首字母改为小写即可。

（1）访问 request 对象

通过 pageContext 获取 JSP 内置对象中的 request 对象，可以使用下面的语句：

```
${pageContext.request}
```

获取 request 对象后，即可通过该对象获取与客户端相关的信息。例如，HTTP 报头信息、客户信息提交方式、客户端主机 IP 地址和端口号等。在 5.2.3 节的表 5.1 中列出了 request 对象用于获取客户端相关信息的常用方法，在此处只需要将方法名中的 get 去掉，并将方法名的首字母改为小写即可。例如，要访问 getServerPort() 方法可以使用下面的代码：

```
${pageContext.request.serverPort }
```

上述代码将返回端口号，这里为 8080。

说明　不可通过 pageContext 对象获取保存在 request 范围内的变量。

（2）访问 response 对象

通过 pageContext 获取 JSP 内置对象中的 response 对象可以使用下面的代码：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
${pageContext.response}
```

获取 response 对象后，即可通过该对象获取与响应相关的信息，例如，响应的内容类型。要获取响应的内容类型，可以使用下面的代码：

```
${pageContext.response.contentType }
```

上述代码将返回响应的内容类型，这里为“text/html;charset=UTF-8”。

（3）访问 out 对象

通过 pageContext 获取 JSP 内置对象中的 out 对象可以使用下面的代码：

```
${pageContext.out}
```

获取 out 对象后，即可通过该对象获取与输出相关的信息。例如，输出缓冲区的大小可以使用下面的代码：

```
${pageContext.out.bufferSize }
```

上述代码将返回输出缓冲区的大小，这里为“8192”。

（4）访问 session 对象

通过 pageContext 获取 JSP 内置对象中的 session 对象可以使用下面的代码：

```
${pageContext.session}
```

获取 session 对象后，即可通过该对象获取与 session 相关的信息。例如，要获取 session 的有效时间可以使用下面的代码：

```
${pageContext.session.maxInactiveInterval}
```

上述代码将返回 session 的有效时间，这里为 1800 秒，即 30 分钟。

（5）访问 exception 对象

通过 pageContext 获取 JSP 内置对象中的 exception 对象可以使用下面的代码：

```
${pageContext.exception}
```

获取 session 对象后，即可通过该对象获取 JSP 页面的异常信息。例如，获取异常信息字符串，可以使用下面的代码：

```
${pageContext.exception.message}
```

说明 *在使用该对象时，也需要在可能出现错误的页面中指定错误处理页，并且在其中指定 page 指令的 isErrorPage 属性值为 true，然后使用上面的 EL 输出异常信息。*

（6）访问 page 对象

通过 pageContext 获取 JSP 内置对象中的 page 对象可以使用下面的代码：

```
${pageContext.page}
```

获取 page 对象后，即可通过该对象获取当前页面的类文件，代码如下：

```
${pageContext.page.class}
```

上述代码将返回当前页面的类文件，这里为 class org.apache.jsp.index_jsp。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（7）访问 servletContext 对象

通过 pageContext 获取 JSP 内置对象中的 servletContext 对象可以使用下面的代码：

```
${pageContext.servletContext}
```

获取 servletContext 对象后，即可通过该对象获取 servlet 上下文信息。例如，获取 servlet 上下文路径的代码如下：

```
${pageContext.servletContext.contextPath}
```

上述代码将返回当前页面的上下文路径，这里为“/8.3”。

8.5.2　访问作用域范围的隐含对象

在 EL 中提供了 4 个用于访问作用域范围的隐含对象，即 pageScope、requestScope、sessionScope 和 applicationScope。应用这 4 个隐含对象指定要查找标识符的作用域后，系统将不再按照默认的顺序（page、request、session 及 application）来查找相应的标识符。它们与 JSP 中的 page、request、session 及 application 内置对象类似，只不过这 4 个隐含对象只能用来取得指定范围内的属性值，而不能取得其他相关信息。

（1）pageScope 隐含对象

pageScope 隐含对象用于返回包含 page（页面）范围内的属性值的集合，返回值为 java.util.Map 对象。下面通过一个具体的例子介绍 pageScope 隐含对象的应用。

【例 8.4】　通过 pageScope 隐含对象读取 page 范围内的 JavaBean 的属性值。

实例位置：光盘 \MR\ 源码 \ 第 8 章 \8.4

Step 1）创建一个名称为 UserInfo 的 JavaBean，并将其保存到 com.wgh 包中。在该 JavaBean 中包括一个 name 属性，具体代码如下：

```
package com.wgh;
public class UserInfo {
    private String name = "";                                    // 用户名
    // name 属性对应的 Setter 方法
    public void setName(String name) {
            this.name = name;
    }
    // name 属性对应的 Getter 方法
    public String getName() {
            return name;
    }
}
```

Step 2）编写 index.jsp 文件，在该文件中应用 <jsp:useBean> 动作标识，创建一个 page 范围内的 JavaBean 实例，并设置 name 属性的值为“琦琦”，具体代码如下：

```
<jsp:useBean id="user" scope="page" class="com.wgh.UserInfo"
    type="com.wgh.UserInfo">
    <jsp:setProperty name="user" property="name" value="琦琦"/>
</jsp:useBean>
```

Step 3）在 index.jsp 的 <body> 标记中，应用 pageScope 隐含对象获取该 JavaBean 实例的 name 属性，代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
用户名为：${pageScope.user.name}
```

运行本实例，将显示如图 8.7 所示的运行结果。



图 8.7　运行结果

（2）requestScope 隐含对象

requestScope 隐含对象用于返回包含 request（请求）范围内的属性值的集合，返回值为 java.util.Map 对象。例如，要获取保存在 request 范围内的 userName 变量，可以使用下面的代码：

```
<%request.setAttribute("userName","mr"); //定义 request 范围内的变量 userName%>
${requestScope.userName}
```

（3）sessionScope 隐含对象

sessionScope 隐含对象用于返回包含 session（会话）范围内的属性值的集合，返回值为 java.util.Map 对象。例如，要获取保存在 session 范围内的 manager 变量，可以使用下面的代码：

```
<%session.setAttribute("manager","mr");          //定义 session 范围内的变量 manager%>
${sessionScope.manager}
```

（4）applicationScope 隐含对象

applicationScope 隐含对象用于返回包含 application（应用）范围内的属性值的集合，返回值为 java.util.Map 对象。例如，要获取保存在 application 范围内的 message 变量，可以使用下面的代码：

```
<%
//定义 application 范围内的变量 message
application.setAttribute("message","欢迎光临无语阁聊天室！");
%>
${applicationScope.message}
```

8.5.3　访问环境信息的隐含对象

在 EL 中，提供了 6 个访问环境信息的隐含对象。下面将对这 6 个隐含对象进行详细介绍。

（1）param 对象

param 对象用于获取请求参数的值，应用在参数值只有一个的情况。在应用 param 对象时，返回的结果为字符串。

例如，在 JSP 页面中，放置一个名称为 name 的文本框，关键代码如下：

```
<input name="name" type="text">
```

当表单提交后，要获取 name 文本框的值，可以使用下面的代码：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
${param.name}
```

注意　如果 name 文本框中可以输入中文，那么在应用 EL 输出其内容前，还需应用 request.setCharacterEncoding("UTF-8"); 语句设置请求的编码为支持中文的编码，否则将产生乱码。

（2）paramValues 对象

如果一个请求参数名对应多个值时，则需要使用 paramValues 对象获取请求参数的值。在应用 paramValues 对象时，返回的结果为数组。

【例 8.5】 应用 paramValues 对象获取复选框组的值。

实例位置：光盘 \MR\ 源码 \ 第 8 章 \8.5

Step 1) 在 JSP 页面中，放置一个名称为 affect 的复选框组，关键代码如下：

```
<input name="affect" type="checkbox" id="affect" value="登山">登山
<input name="affect" type="checkbox" id="affect" value="游泳">游泳
<input name="affect" type="checkbox" id="affect" value="慢走">慢走
<input name="affect" type="checkbox" id="affect" value="晨跑">晨跑
```

Step 2) 当表单提交后，要获取 affect 的值，可以使用下面的代码：

```
<%request.setCharacterEncoding("UTF-8"); %>
爱好为：${paramValues.affect[0]}${paramValues.affect[1]}${paramValues.
affect[2]}${paramValues.affect[3]}
```

注意　在应用 param 和 paramValues 对象时，如果指定的参数不存在，则返回空的字符串，而不是返回 null。

运行本实例，将显示如图 8.8 所示的运行结果。



图 8.8　应用 paramValues 对象获取复选框的值

（3）header 和 headerValues 对象

header 对象用于获取 HTTP 请求的一个具体的 header 的值，但是在有些情况下，可能存在同一个 header 拥有多个不同的值，这时就必须使用 headerValues 对象。

例如，要获取 HTTP 请求的 header 的 connection（是否需要持久连接）属性，可以应用以下代码实现。

```
${header.connection}或${header["connection"]}
```

上面的 EL 表达式将输出如图 8.9 所示的结果。



图 8.9　应用 header 对象获取的 connection 属性

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

但是，如果要获取 HTTP 请求的 header 的 user-agent 属性，则必需应用以下 EL 表达式。

```
${header["user-agent"]}
```

上面的代码将输出如图 8.10 所示的结果。

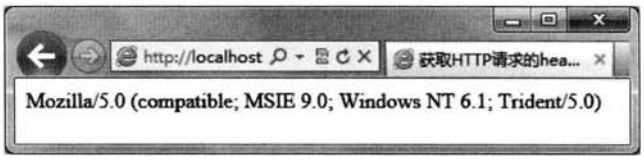


图 8.10　应用 header 对象获取的 user-agent 属性

（4）initParam 对象

initParam 对象用于获取 Web 应用初始化参数的值。例如，在 Web 应用的 web.xml 文件中设置一个初始化参数 author，用于指定作者，具体代码如下：

```
<context-param>
   <param-name>author</param-name>
   <param-value>mr</param-value>
</context-param>
```

应用 EL 获取参数 author 的代码如下：

```
${initParam.author}
```

【例 8.6】 获取 Web 应用初始化参数并显示。

实例位置：光盘 \MR\ 源码 \ 第 8 章 \8.6

Step 1) 打开 web.xml 文件，在 </web-app> 标记的上方添加以下设置初始化参数的代码。

```
<context-param>
   <param-name>company</param-name>
   <param-value> 吉林省明日科技有限公司 </param-value>
</context-param>
```

在上面的代码中，设置了一个名称为 company 的参数，参数值为“吉林省明日科技有限公司”。

Step 2) 编写 index.jsp 文件，在该文件中应用 EL 获取并显示初始化参数 company，关键代码如下：

```
版权所有：${initParam.company}
```

运行本实例，将显示如图 8.11 所示的运行结果。

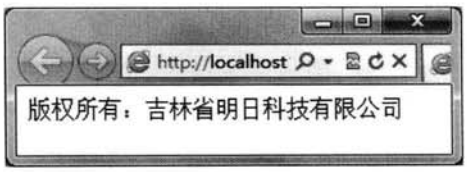


图 8.11　运行结果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

❑ cookie 对象

虽然在 EL 中，并没有提供向 cookie 中保存值的方法，但是它提供了访问由请求设置的 cookie 的方法，这可以通过 cookie 隐含对象实现。如果在 cookie 中已经设定一个名称为 username 的值，那么可以使用 ${cookie.username} 来获取该 cookie 对象。但是如果要获取 cookie 中的值，需要使用 cookie 对象的 value 属性。

例如，使用 response 对象设置一个请求有效的 cookie 对象，然后再使用 EL 获取该 cookie 对象的值，可以使用下面的代码：

```
<%Cookie cookie=new Cookie("user","mrbccd");
  response.addCookie(cookie);
%>
${cookie.user.value}
```

运行上面的代码后，将在页面中显示 mrbccd。

说明　所谓的 cookie 是一个文本文件，它是以 key、value 的方法将用户会话信息记录在这个文本文件内，并将其暂时存放在客户端浏览器中。

8.6　定义和使用EL函数

在 EL 中允许定义和使用函数，本节将介绍如何定义和使用 EL 函数，以及可能出现的错误。

8.6.1　定义和使用函数

定义和使用函数包括以下 3 个步骤。

Step 1）编写一个 Java 类，并在其中编写公用的静态方法，用于实现自定义 EL 函数的具体功能。

Step 2）编写标签库描述文件声明函数，该文件的扩展名为“.tld”，保存在 Web 应用的 WEB-INF 文件夹下。

Step 3）在 JSP 页面中引用标签库，并调用定义的 EL 函数实现相应的功能。

下面将通过一个具体的实例介绍 EL 函数的定义和使用。

【例 8.7】 定义 EL 函数处理字符串中的回车换行符和空格符。

实例位置：光盘 \MR\ 源码 \ 第 8 章 \8.7

Step 1）编写一个 Java 类，名为“StringDeal”并保存在 com.wgh 包中。在其中添加一个公用的静态方法 shiftEnter()，用于将输入字符串中的回车换行符替换为“
”，将空格符替换为“ ”，然后返回新替换后的字符串，其完整代码如下：

```
package com.wgh;
public class StringDeal {
   public static String shiftEnter(String str) {                    // 定义公用的静态方法
          String newStr = str.replaceAll("\r\n", "<br>");            // 替换回车换行符
          newStr = newStr.replaceAll(" ", " ");                 // 替换空格符
          return newStr;
   }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Step 2) 编写标签库描述文件，名为“stringDeal.tld”，并保存在 WEB-INF 文件夹下，其关键代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<taglib xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.
w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
   web-jsptaglibrary_2_0.xsd"
   version="2.0">
   <tlib-version>1.0</tlib-version>
   <uri>/stringDeal</uri>
   <function>
          <name>shiftEnter</name>
          <function-class>com.wgh.StringDeal</function-class>
          <function-signature>java.lang.String shiftEnter(java.lang.String)
          </function-signature>
   </function>
</taglib>
```

<uri> 标记：用于指定 tld 文件的映射路径，在应用 EL 函数时需要使用该标记指定的内容。

<name> 标记：用于指定 EL 函数所对应方法的方法名，通常与 Java 文件中的方法名相同。

<function-class> 标记：用于指定 EL 函数所对应的 Java 文件，其中包括包名和类名。例如，在上面的代码中包名为 com.wgh，类名为 StringDeal。

<function-signature> 标记：用于指定 EL 函数所对应的静态方法，其中包括返回值和入口参数的类型，在指定这些类型时需要使用完整的类型名。例如，在上面的代码中不能指定该标记的内容为 String shiftEnter(String)。

Step 3) 编写 index.jsp 文件，在其中添加一个表单及表单元素用于收集内容信息，关键代码如下：

```
<form name="form1" method="post" action="deal.jsp">
  <textarea name="content" cols="30" rows="5"></textarea>
  <br>
  <input type="submit" name="Button" value="提交" >
</form>
```

Step 4) 编写表单的处理页 deal.jsp 文件，在其中应用上面定义的 EL 函数。处理获取的内容（主要是替换字符串中的回车换行符和空格符）后显示在页面中。deal.jsp 文件的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib uri="/stringDeal" prefix="wghfn"%>
<%request.setCharacterEncoding("UTF-8"); %>
… <!-- 此处省略了部分 HTML 代码 -->
内容为：<br>
${wghfn:shiftEnter(param.content)}
</body>
</html>
```

说明 在引用标签库时指定的 uri 属性与标签库描述文件中的 <uri> 标记值必须相对应。

运行本实例，在页面中显示一个内容编辑框和一个“提交”按钮。在内容编辑框中输入

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

如图 8.12 所示的内容，单击“提交”按钮，显示如图 8.13 所示的运行结果。

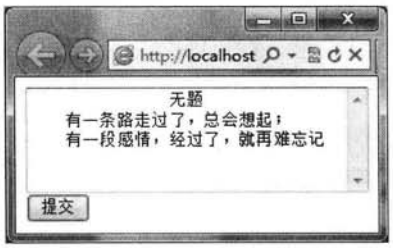


图 8.12　输入文本

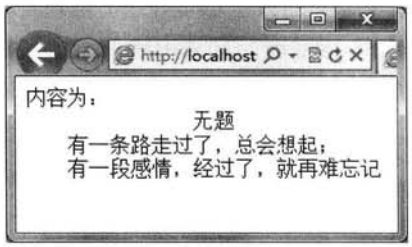


图 8.13　获取结果

8.6.2　定义和使用 EL 函数时常见的错误

在定义和使用 EL 函数时可能出现以下 3 种错误。

（1）由于没有指定完整的类型名而产生的异常信息

在编写 EL 函数时，如果出现如图 8.14 所示的异常信息，则是由于在标签库描述文件中没有指定完整的类型名而产生的。



图 8.14　由于没有指定完整的类型名而产生的异常信息

解决的方法是在扩展名为“.tld”文件中指定完整的类型名，如在上面的这个异常中，即可将完整的类型名设置为 java.lang.String。

（2）由于在标签库的描述文件中输入了错误的标记名产生的异常信息

在编写 EL 函数时，如果出现如图 8.15 所示的异常信息，则可能是由于在标签库描述文件中输入了错误的标记名造成的。例如，这个异常信息就是由于将标记名“<function-signature>写为 <function-signatrue>”所导致的。



图 8.15　由于在标签库的描述文件中输入了错误的标记名产生的异常信息

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

解决的方法是将错误的标记名修改正确，并重新启动服务器运行程序。

（3）由于定义的方法不是静态方法所产生的异常信息

在编写 EL 函数时，如果出现如图 8.16 所示的异常信息，则可能是由于在编写 EL 函数时使用的 Java 类中定义的函数对应的方法不是静态方法所造成的。



图 8.16　由于定义的方法不是静态方法所产生的异常信息

解决的方法是将该方法修改为静态方法，即在声明方法时使用 static 关键字。

8.7　上机实践

8.7.1　应用 EL 访问 JavaBean 属性

➤➤➤ 题目描述

本上机实践的主要功能是把用户填写的表单数据存放在 JavaBean 中，在提交的处理页面中通过 EL 表达式读取 JavaBean 中的数据并将其显示在网页中。例如，在 index.jsp 页面中，用户输入的基本信息如图 8.17 所示。当用户单击“提交”按钮后打开 getDate.jsp 页面文件，该页面显示用户输入的用户信息，如图 8.18 所示。



图 8.17　用户注册页面

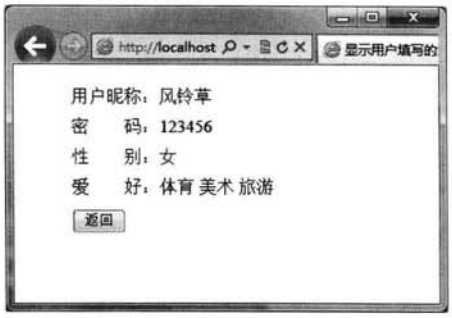


图 8.18　显示用户输入的信息

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

➤➤➤ 技术指导

本上机实践的关键开发步骤如下。

Step 1) 创建名为 UserBean.java 的 JavaBean 文件，主要用于声明用户的基本信息属性，并且设置 Getter 和 Setter 方法。

Step 2) 创建名为 index.jsp 的页面文件，用于添加用户注册所需要的表单信息。

Step 3) 创建 getDate.jsp 页面文件，在该文件中，首先应用 request 内置对象的 setCharacterEncoding() 方法设置请求的编码的方式为 UTF-8，然后应用 <jsp:useBean> 动作指令在页面中创建一个 JavaBean 实例，再应用 <jsp:setProperty> 动作指令设置 JavaBean 实例的各属性值，最后应用 EL 表达式将 JavaBean 的各属性显示到页面中。该页面的关键代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%request.setCharacterEncoding("UTF-8");%>
<jsp:useBean id="userForm" class="com.wgh.UserForm" scope="page"/>
<jsp:setProperty name="userForm" property="*"/>
… <!-- 此处省略了部分 HTML 和 CSS 代码 -->
<ul>
   <li>用户昵称：${userForm.username}</li>
   <li>密　　码：${userForm.pwd}</li>
    <li>性　　别：${userForm.sex}</li>
      <li>爱　　好：${userForm.affect[0]} ${userForm.affect[1]} ${userForm.
      affect[2]} ${userForm.affect[3]}</li>
      <li><input name="Button" type="button" class="btn_grey" value=" 返回 "
 onClick="window.location.href='index.jsp'"></li>
</ul>
```

8.7.2　应用 EL 表达式显示投票结果

➤➤➤ 题目描述

本上机实践要求编写一个程序，实现应用 EL 表达式显示投票结果。运行效果如图 8.19 和图 8.20 所示。在如图 8.19 所示的投票页面中，选中自己需要的编程类图书，单击“投票”按钮，将完成投票，并显示投票结果，如图 8.20 所示。在投票页面，单击“查看投票结果”按钮也可以查看投票结果。



图 8.19　投票页面



图 8.20　显示投票结果页面

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

➤➤➤ 技术指导

在实现本实例时，首先需要创建一个用于添加收集投票信息的表单及表单元素的 JSP 页面，然后编写一个完成投票功能的 Servlet，最后再编写一个用于应用 EL 表达式输出投票结果的 JSP 页面。本上机实践的关键代码如下：

```
基础教程类:  <img src="bar.gif" width='${220*(applicationScope.pollResult["基
础教程类"]/(applicationScope.pollResult["基础教程类"]+applicationScope.pollResult["实例
集锦类"]+applicationScope.pollResult["经验技巧类"]+applicationScope.pollResult["速查手册
类"]+applicationScope.pollResult["案例剖析类"]))}' height="13">
        (${empty applicationScope.pollResult["基础教程类"]? 0 :applicationScope.
pollResult["基础教程类"]})
```

说明 这里只给出了显示第一个投票项的关键代码，其他投票项的代码与其类似。

8.7.3 应用 EL 显示用户的登录状态

➤➤➤ 题目描述

本上机实践要求编写一个程序，实现应用 EL 表达式显示用户的登录状态。当用户没有登录时，显示“您还没有登录”文字和“登录”超链接；当用户登录后，显示登录的用户名和“退出”超链接。运行效果如图 8.21 和图 8.22 所示。



图 8.21 没有登录的显示结果



图 8.22 登录后显示的运行结果

➤➤➤ 技术指导

在实现本实例时，可以应用 EL 表达式提供的条件运算符和 empty 运算符来实现。条件运算符语法格式如下：

```
${ 条件表达式 ? 表达式 1 : 表达式 2}
```

- 条件表达式：用于指定一个条件表达式，其值为 boolean 型，可以由关系运算符、逻辑运算符和 empty 运算符组成。
- 表达式 1：用于指定当条件表达式的值为 true 时，将要返回的值。
- 表达式 2：用于指定当条件表达式的值为 false 时，将要返回的值。

本上机实践的关键代码如下：

```
${empty sessionScope.user ? "您还没有登录<a href='login.jsp'>登录</a>" :
sessionScope.user}
${!empty sessionScope.user ? "欢迎您来到本网站[<a href='logout.jsp'>退出</a>]":""}
```

8.8 小结

本章首先简要介绍了表达式语言 EL、EL 的基本语法及特点，然后详细介绍了禁用 EL 的

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

3 种方法、EL 的保留关键字、运算符及其优先级和隐含对象，最后介绍了如何定义和使用 EL 的函数，以及定义和使用 EL 函数时可能出现的异常信息及其解决方法。

8.9　习题

选择题

1．编程时禁用 EL 表达式的方式是（　　）。

A. 使用 \　　B. 使用 /* */　　C. 使用 <!-- -->

2. 下面不是 EL 表达式特点的是（　　）。

A. 访问 JavaBean 属性　　B. 访问 JSP 作用域　　C. 任何浏览器都支持

3．不是存取数据运算符的是（　　）。

A. .　　B. []　　C. ()

4．EL 表达式在对隐含对象进行查找时最先查找的是（　　）。

A. session　　B. page　　C. application

5．EL 表达式中运算符优先级别最高的是（　　）。

A. ()　　B. []　　C. !

填空题

1．param 对象用于获取请求参数的值，而如果一个参数名对应多个值时，则需要使用_____对象获取请求参数的值。

2．所谓的 cookie 是一个文本文件，它是以_____的方法将用户会话信息记录在这个文本文件内，并将其暂时存放在客户端浏览器中。

3．写出 EL 表达式中条件运算符的基本语法格式_____。

4．要获取 session 范围内的 user 变量的值，可以用的 EL 表达式是_____。

5．禁用 EL 表达式的三种方法是_____、_____、_____。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 9 章　页面控制利器——JSTL 标签库

视频讲解：85 分钟

JSTL（JavaServer Pages Standard Tag Library，JSP 标准标签库）由 Apache 的 Jakarta 小组负责维护，它是一个不断完善的开放源代码的 JSP 标准标签库，主要为 Java Web 开发人员提供一个标准且通用的标签库。使用 JSTL 可以取代在 JSP 程序中嵌入 Java 代码的做法，大大提高程序的可维护性。本章将对 JSTL 的下载和配置以及 JSTL 的核心标签进行详细介绍。

通过学习本章，读者可以达到以下学习目的：

- 了解如何引用 JSTL 提供的各种标签库；
- 掌握 JSTL 的下载与配置；
- 掌握如何使用 JSTL 提供的表达式标签；
- 掌握如何使用 JSTL 提供的 URL 相关的标签；
- 掌握如何使用 JSTL 提供的流程控制标签；
- 掌握如何使用 JSTL 提供的循环标签。

9.1　JSTL标签库简介

虽然 JSTL 叫做标准标签库，实际上它是由 5 个功能不同的标签库组成。这 5 个标签库分别是核心标签库、格式标签库、SQL 标签库、XML 标签库和函数标签库。在使用这些标签之前必须在 JSP 页面的顶部使用 <%@ taglib%> 指令定义引用的标签库和访问前缀。

使用核心标签库的 taglib 指令格式如下：

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

使用格式标签库的 taglib 指令格式如下：

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
```

使用 SQL 标签库的 taglib 指令格式如下：

```
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>
```

使用 XML 标签库的 taglib 指令格式如下：

```
<%@ taglib prefix="xml" uri="http://java.sun.com/jsp/jstl/xml"%>
```

使用函数标签库的 taglib 指令格式如下：

```
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
```

下面我们就来对 JSTL 提供的这 5 个标签库分别进行简要介绍。

（1）核心标签库

核心标签库主要用于完成 JSP 页面的常用功能，包括 JSTL 的表达式标签、URL 标签、流

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

程控制标签和循环标签共 4 种标签。其中，表达式标签包括 <c:out>、<c:set>、<c:remove> 和 <c:catch>；URL 标签包括 <c:import>、<c:redirect>、<c:url> 和 <c:param>；流程控制标签包括 <c:if>、<c:choose>、<c:when> 和 <c:otherwise>；循环标签包括 <c:forEach> 和 <c:forTokens>。这些标签的基本作用如表 9.1 所示。

表 9.1　核心标签库

标　签	说　明
<c:out>	将表达式的值输出到 JSP 页面中，相当于 JSP 表达式 <% = 表达式 %>
<c:set>	在指定范围中定义变量，或为指定的对象设置属性值
<c:remove>	从指定的 JSP 范围中移除指定的变量
<c:catch>	捕获程序中出现的异常，相当于 Java 语言中的 try...catch 语句
<c:import>	导入站内或其他网站的静态和动态文件到 Web 页面中
<c:redirect>	将客户端发出的 request 请求重定向到其他 URL 服务端
<c:url>	使用正确的 URL 重写规则构造一个 URL
<c:param>	为其他标签提供参数信息，通常与其标签结合使用
<c:if>	根据不同的条件去处理不同的业务，与 Java 语言中的 if 语句类似，只不过该语句没有 else 标签
<c:choose>、<c:when>、<c:otherwise>	根据不同的条件去完成指定的业务逻辑，如果没有符合的条件，则会执行默认条件的业务逻辑，相当于 Java 语言中的 switch 语句
<c:forEach>	根据循环条件，遍历数组和集合类中的所有或部分数据
<c:forTokens>	迭代字符串中由分隔符分隔的各成员

（2）格式标签库

格式标签库提供了一个简单的国际化标记，也被称为 I18N 标签库，用于处理和解决国际化相关的问题，另外，格式标签库中还包含用于格式化数字和日期显示格式的标签。由于该标签库在实际项目开发中并不经常应用，这里不做详细介绍。

（3）SQL 标签库

SQL 标签库提供了基本的访问关系型数据的能力。使用 SQL 标签，可以简化对数据库的访问。如果结合核心标签库，可以方便地获取结果集，并迭代输出结果集中的数据。由于该标签库在实际项目开发中并不经常应用，这里不做详细介绍。

（4）XML 标签库

XML 标签库可以处理和生成 XML 的标记，使用这些标记可以很方便地开发基于 XML 的 Web 应用。由于该标签库在实际项目开发中并不经常应用，这里不做详细介绍。

（5）函数标签库

函数标签库提供了一系列字符串操作函数，用于完成分解字符串、连接字符串、返回子串、确定字符串是否包含特定的子串等功能。由于该标签库在实际项目开发中并不经常应用，这里不做详细介绍。

9.2　JSTL的下载与配置

由于 JSTL 还不是 JSP 规范中的一部分，所以在使用 JSTL 之前，需要安装并配置 JSTL。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

下面将介绍如何下载与配置 JSTL。

9.2.1 下载 JSTL 标签库

JSTL 标签库可以到 http://jstl.java.net/download.html 网站中下载。在该页面中，将提供两个超链接，一个是 JSTL API 超链接（用于下载 JSTL 的 API），另一个是 JSTL Implementation 超链接（用于下载 JSTL 的实现 Implementation）。单击 JSTL API 超链接下载 JSTL 的 API，下载后的文件名为 javax.servlet.jsp.jstl-api-1.2.1.jar；单击 JSTL Implementation 超链接下载 JSTL 的实现 Implementation，下载后的文件名为 javax.servlet.jsp.jstl-1.2.1.jar。

9.2.2 配置 JSTL

JSTL 的标签库下载完毕后，就可以在 Web 应用中配置 JSTL 标签库。配置 JSTL 标签库有两种方法，一种是直接将 javax.servlet.jsp.jstl-api-1.2.1.jar 和 javax.servlet.jsp.jstl-1.2.1.jar 复制到 Web 应用的 WEB-INF\lib 目录中即可；另一种是在 Eclipse 中通过配置构建路径的方法进行添加。在 Eclipse 中通过配置构建路径的方法添加 JSTL 标签库的具体步骤如下。

Step 1 在项目名称节点上，单击鼠标右键，在弹出的快捷菜单中选择“构建路径”/“配置构建路径”菜单项，将打开“Java 构建路径”对话框，在该对话框中，单击“添加库”按钮，将打开“添加库”对话框，选择“用户库”节点，单击“下一步”按钮，将打开如图 9.1 所示的对话框。

Step 2 单击“用户库”按钮，将打开有关首选项的对话框，在该对话框中，单击“新建”按钮，将打开“新建用户库”对话框，在该对话框中输入用户库名称，这里为 JSTL 1.2.1，如图 9.2 所示。



图 9.1 “添加库”对话框

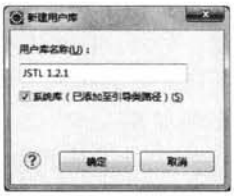


图 9.2 “新建用户库”对话框

Step 3 单击“确定”按钮，返回到“首选项（已过滤）”对话框，在该对话框中将显示刚刚创建的用户库，如图 9.3 所示。



图 9.3 “首选项（已过滤）”对话框

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Step 4 选中 JSTL 1.2.1 节点，单击“添加 JAR”按钮，在打开的“选择 JAR”对话框中，选择刚刚下载的 JSTL 标签库，如图 9.4 所示。



图 9.4　选择 JSTL 标签库

Step 5 单击“打开”按钮，将返回到如图 9.5 所示的“首选项（已过滤）”对话框中。



图 9.5　添加 Jar 后的“首选项（已过滤）”对话框

Step 6 单击“确定”按钮，返回到“添加库”对话框，在该对话框中，单击“完成”按钮，完成 JSTL 库的添加。选中当前项目，并刷新该项目，这时依次展开如图 9.6 所示的节点，可以看到在项目节点下，将添加一个 JSTL 1.2.1 节点。

Step 7 在项目名称节点上，单击鼠标右键，在弹出的快捷菜单中选择“属性”菜单项，将打开项目属性对话框，在该对话框的左侧列表中，选择 Deployment Assembly 节点，在右侧将显示 Web Deployment Assembly 信息，单击“添加”按钮，将打开如图 9.7 所示的对话框。



图 9.6　添加到 Eclipse 项目中的 JSTL 库

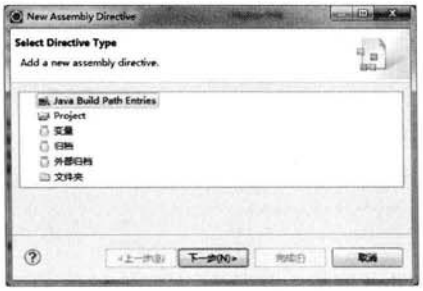


图 9.7　New Assembly Directive 对话框

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Step 8 双击 Java Build Path Entries 列表项，将显示如图 9.8 所示的对话框。



图 9.8 选择用户库对话框

Step 9 在该对话框中，选择要添加的用户库，单击“完成”按钮。即可将该用户库添加到 Web Deployment Assembly 中，添加后的效果如图 9.9 所示。

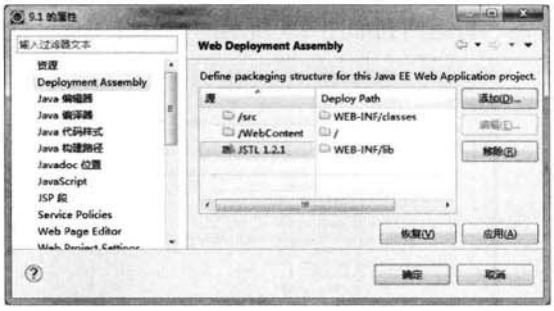


图 9.9 添加用户库到 Web Deployment Assembly 中的效果

说明 这里介绍的添加 JSTL 标签库文件到项目中的方法，也适用于添加其他的库文件。

至此，下载并配置 JSTL 的基本步骤就完成了。这时就可以在项目中使用 JSTL 标签库了。

9.3 表达式标签

9.3.1 <c:out> 输出标签

<c:out> 标签用于将表达式的值输出到 JSP 页面中，该标签类似于 JST 的表达式 <%= 表达式 %>，或者 EL 表达式 ${expression}。<c:out> 标签有两种语法格式，一种没有标签体，另一种有标签体，这两种语法的输出结果完全相同。<c:out> 标签的具体语法格式如下：

语法 1——没有标签体：

```
<c:out value="expression" [escapeXml="true|false"] [default="defaultValue"]/>
```

语法 2——有标签体：

```
<c:out value="expression" [escapeXml="true|false"]>
   defalultValue
</c:out>
```

- value 属性：用于指定将要输出的变量或表达式。该属性值的类似为 Object，可以使用 EL。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

❑ escapeXml 属性：可选属性，用于指定是否转换特殊字符，可以被转换的字符如表 9.2 所示。其属性值为 true 或 false，默认值为 true，表示转换。例如，可以将“<”转换为“<”。

表 9.2　被转换的字符

字　符	字符实体代码	字　符	字符实体代码
<	<	>	>
'	'	"	"
&	&		

❑ default 属性：可选属性，用于指定当 value 属性值等于 null 时，将要显示的默认值。如果没有指定该属性，并且 value 属性的值为 null，该标签将输出空的字符串。

【例 9.1】 应用 <c:out> 标签输出字符串“水平线标记 <hr>”。

实例位置：光盘 \MR\ 源码 \ 第 9 章 \9.1

编写 index.jsp 文件，在该文件中，首先应用 taglib 指令引用 JSTL 的核心标签库，然后添加两个 <c:out> 标签，用于输出字符串“水平线标记 <hr>”，这两个 <c:out> 标签的 escapeXml 属性的值分别为 true 和 false。index.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>应用 &lt;c:out&gt; 标签输出字符串 " 水平线标记 &lt;hr&gt;"</title>
</head>
<body>
escapeXml 属性为 true 时：
<c:out value=" 水平线标记 <hr>" escapeXml="true"></c:out>
<br>
escapeXml 属性为 false 时：
<c:out value=" 水平线标记 <hr>" escapeXml="false"></c:out>
</body>
</html>
```

运行本实例，将显示如图 9.10 所示的运行结果。

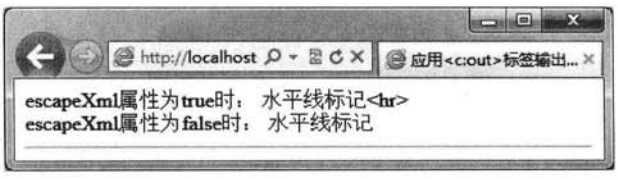


图 9.10　运行结果

从图 9.10 中，我们可以看出当 scapeXml 属性值为 true 时，输出字符串中的 <hr> 被以字符串的形式输出了，而当 scapeXml 属性值为 false 时，字符串中的 <hr> 则被当做 HTML 标记进行输出。这是因为，当 scapeXml 属性值为 true 时，已经将字符串中的 < 和 > 符号转换为对

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

应的实体代码，所以在输出时，就不会被当作 HTML 标记进行输出了。这一点，可以通过查看源代码看出。本实例在运行后，将得到下面的源代码。

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>应用 &lt;c:out&gt; 标签输出字符串 " 水平线标记 &lt;hr&gt;"</title>
</head>
<body>
escapeXml 属性为 true 时：
水平线标记 &lt;hr&gt;
<br>
escapeXml 属性为 false 时：
水平线标记 <hr>
</body>
</html>
```

9.3.2 <c:set> 变量设置标签

<c:set> 标签用于在指定范围（page、request、session 或 application）中定义保存某个值的变量，或为指定的对象设置属性值。使用该标签可以在页面中定义变量，而不用在 JSP 页面中嵌入打乱 HTML 排版的 Java 代码。<c:set> 标签有 4 种语法格式。

语法 1：在 scope 指定的范围内将变量值存储到变量中。

```
<c:set var="name" value="value" [scope=" 范围 "]/>
```

语法 2：在 scope 指定的范围内将标签体存储到变量中。

```
<c:set var="name" [scope="page|request|session|application"]>
   标签体
</c:set>
```

语法 3：将变量值存储在 target 属性指定的目标对象的 propName 属性中。

```
<c:set value="value" target="object" property="propName"/>
```

语法 4：将标签体存储到 target 属性指定的目标对象的 propName 属性中。

```
<c:set target="object" property="propName">
   标签体
</c:set>
```

<c:set> 标签的属性说明如表 9.3 所示。

表 9.3 <c:set> 标签的属性说明

属　性	说　明
var	用于指定变量名。通过该标签定义的变量名，可以通过 EL 指定为 <c:out> 的 value 属性的值
value	用于指定变量值，可以使用 EL
scope	用于指定变量的作用域，默认值是 page。可选值包括 page、request、session 和 application
target	用于指定存储变量值或者标签体的目标对象，可以是 JavaBean 或 Map 集合对象
property	用于指定目标对象存储数据的属性名

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

注意 target 属性不能是直接指定的 JavaBean 或 Map，而应该是使用 EL 表达式或一个脚本表达式指定的真正对象。这个对象可以是通过 <jsp:useBean> 创建的实例。例如，要为 JavaBean “CartForm” 的 id 属性赋值，那么 target 属性值应该是 target="${cart}"，而不应该是 target="cart"。其中 cart 为 CartForm 的对象。

【例 9.2】 应用 <c:set> 标签定义变量和为 JavaBean 属性赋值。

实例位置：光盘 \MR\ 源码 \ 第 9 章 \9.2

Step 1 编写一个名称为 UserInfo 的 JavaBean，并保存到 com.wgh 包中。在该 JavaBean 中添加一个 name 属性，并为该属性添加对应的 Setter 和 Getter 方法，具体代码如下：

```
public class UserInfo {
    private String name="";                                    //名称属性
    public void setName(String name) {
            this.name = name;
    }
    public String getName() {
            return name;
    }
}
```

Step 2 编写 index.jsp 文件，在该文件中，首先应用 taglib 指令引用 JSTL 的核心标签库，然后应用 <c:set> 标签定义一个 request 范围内的变量 username，并应用 <c:out> 标签输出该变量，接下来再应用 <jsp:useBean> 动作标识创建 JavaBean 的实例，最后应用 <c:set> 标签为 JavaBean 中的 name 属性设置属性值，并应用 <c:out> 标签输出该属性。index.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>应用 &lt;c:set&gt; 标签的应用 </title>
<style>
li{
padding:5px;
}
</style>
</head>
<body>
<ul>
<li> 定义 request 范围内的变量 username</li>
<c:set var="username" value=" 明日科技 " scope="request"/>
<c:out value="username 的值为：${username}"/>
<li> 设置 UserInfo 对象的 name 属性 </li>
<jsp:useBean class="com.wgh.UserInfo" id="userInfo"/>
<c:set target="${userInfo}" property="name">wgh</c:set>
<c:out value="UserInfo 的 name 属性值为：${userInfo.name}"></c:out>
</ul>
</body>
</html>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

运行本实例，将显示如图 9.11 所示的运行结果。

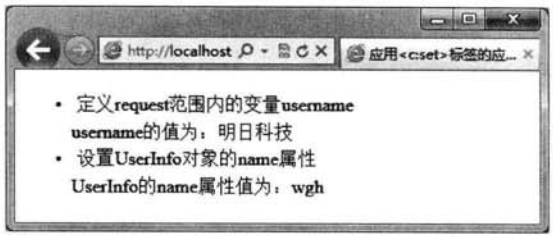


图 9.11　运行结果

在使用语法 3 和语法 4 时，如果 target 属性值为 null、属性值不是 java.util.Map 对象或者不是 JavaBean 对象的有效属性，将抛出如图 9.12 所示的异常。如果读者在程序开发过程中遇到类似的异常信息需要检查 target 属性的值是否合法。

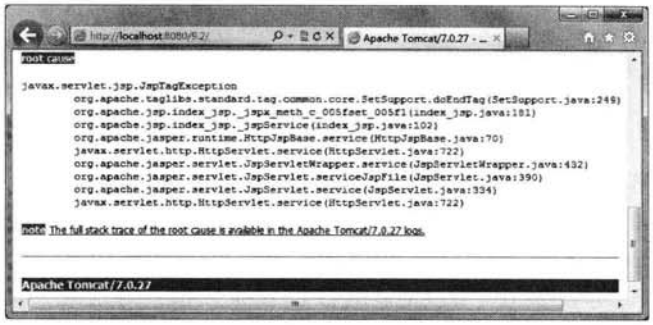


图 9.12　target 属性值不合法时产生的异常

如果在使用 <c:set> 标签的语法 3 和语法 4 时，产生如图 9.13 所示的异常信息，这是因为该标签的 property 属性值指定了一个 Map 对象或是 JavaBean 对象中不存在的属性而产生的。

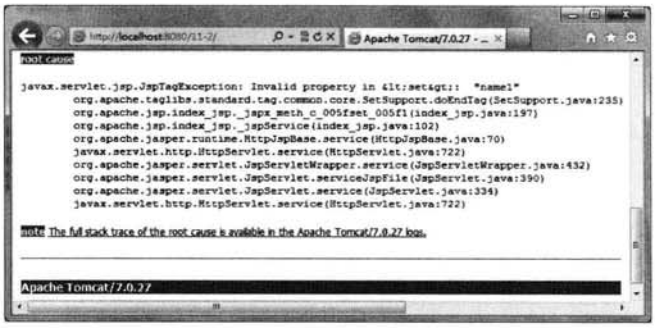


图 9.13　property 属性值不合法时产生的异常

9.3.3　<c:remove> 变量移除标签

<c:remove> 标签用于移除指定的 JSP 范围内的变量，其语法格式如下：

```
<c:remove var="name" [scope="范围"]/>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- var 属性：用于指定要移除的变量名。
- scope 属性：用于指定变量的有效范围，可选值有 page、request、session、application。默认值是 page。如果在该标签中没有指定变量的有效范围，那么将分别在 page、request、session和application的范围内查找要移除的变量并移除。例如，在一个页面中，存在不同范围的两个同名变量，当不指定范围时移除该变量，这两个范围内的变量都将被移除。因此，在移除变量时，最好指定变量的有效范围。

说明　当指定的要移除的变量并不存在时，并不会抛出异常。

【例 9.3】 应用 <c:remove> 标签移除变量。

实例位置：光盘 \MR\ 源码 \ 第 9 章 \9.3

编写 index.jsp 文件，在该文件中，首先应用 taglib 指令引用 JSTL 的核心标签库，然后应用 <c:set> 标签定义一个 request 范围内的变量 username，并应用 <c:out> 标签输出该变量，接下来使用 <c:remove> 标签移除定义的 username 变量，并重新输出。index.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
… < !-- 此处省略了部分 HTML 和 CSS 代码 -->
    <ul>
            <li> 定义 request 范围内的变量 username</li>
            <c:set var="username" value=" 明日科技 " scope="request" />
            username 的值为：
            <c:out value="${username}" />
            <li> 移除 request 范围内的变量 username</li>
            <c:remove var="username" scope="request" />
            username 的值为：
            <c:out value="${username}" default=" 空 " />
    </ul>
</body>
</html>
```

运行本实例，将显示如图 9.14 所示的运行结果。

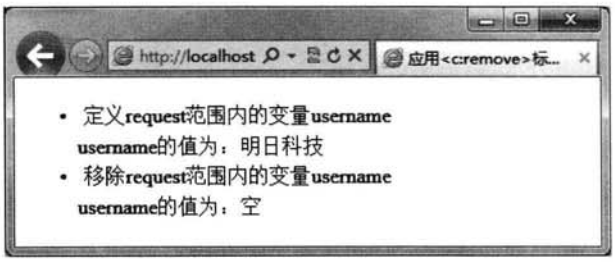


图 9.14　运行结果

9.3.4　<c:catch> 捕获异常标签

<c:catch> 标签用于捕获程序中出现的异常，如果需要它还可以将异常信息保存在指定的变量中。该标签与 Java 语言中的 try...catch 语句类似。<c:catch> 标签的语法格式如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<c:catch [var="exception"]>
…                    // 可能存在异常的代码
</c:catch>
```

- var 属性：可选属性，用于指定存储异常信息的变量。如果不需要保存异常信息，可以省略该属性。

注意 var 属性值只有在 <c:catch> 标签的后面才有效，也就是说，在 <c:catch> 标签体中无法使用有关异常的任何信息。

【例 9.4】 应用 <c:catch> 标签捕获异常信息。

实例位置：光盘 \MR\ 源码 \ 第 9 章 \9.4

Step 1) 编写一个名称为 UserInfo 的 JavaBean，并保存到 com.wgh 包中。在该 JavaBean 中添加一个 name 属性，并为该属性添加对应的 Setter 和 Getter 方法，具体代码如下：

```
package com.wgh;
public class UserInfo {
    private String name="";                                    // 名称属性
    public void setName(String name) {
            this.name = name;
    }
    public String getName() {
            return name;
    }
}
```

Step 2) 编写 index.jsp 文件，在该文件中，首先应用 taglib 指令引用 JSTL 的核心标签库，然后在页面的 <body> 标记中添加 <c:catch> 标签，用于捕获页面中的异常信息，并将异常信息保存到变量 error 中，接下来再在 <c:catch> 标签中应用 <jsp:useBean> 动作标识创建 JavaBean 的实例，并调用 <c:set> 标签为该 JavaBean 的 pwd 属性设置值，最后输出保存异常信息的变量 error。index.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
… <!-- 此处省略了部分 HTML 代码 -->
<c:catch var="error">
    <jsp:useBean class="com.wgh.UserInfo" id="userInfo"/>
    <c:set target="${userInfo}" property="pwd">111</c:set>
</c:catch>
<c:out value="${error}"/>
</body>
</html>
```

运行上面的代码，由于 JavaBean “UserInfo” 中并没有 pwd 属性，所以页面中将显示如图 9.15 所示的异常信息。

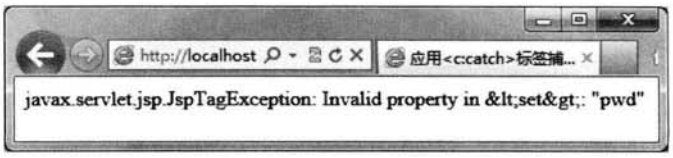


图 9.15 抛出的异常信息

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明 对于本实例，如果想让其不产生异常，可以将 property 属性的属性值改为 name，或者在 JavaBean“UserInfo”中添加 pwd 属性，以及对应的 Setter 方法和 Getter 方法。另外，<c:catch> 标签不仅可以用来获取由其他 JSTL 标签引起的异常信息，而且可以获取页面中由其他的 JSP 脚本标识和动作标识运行时所产生的异常信息，但不能是语法错误。

9.4　URL相关标签

文件导入、重定向和 URL 地址生成是 Web 应用中常用的功能。JSTL 中也提供了与 URL 相关的标签，分别是 <c:import>、<c:redirect>、<c:url> 和 <c:param>。其中，<c:param> 标签通常与其他标签配合使用。

9.4.1　<c:import> 导入标签

<c:import> 标签可以导入站内或其他网站的静态和动态文件到 Web 页面中，例如，使用 <c:import> 标签导入其他网站的天气信息到自己的网页中。<c:import> 标签与 <jsp:include> 动作指令类似，所不同的是 <jsp:include> 只能导入站内资源，而 <c:import> 标签不仅可以导入站内资源，也可以导入其他网站的资源。

<c:import> 标签的语法格式如下：

语法 1：

```
<c:import url="url" [context="context"] [var="name"] [scope="范围"]
[charEncoding="encoding"]>
[标签体]
</c:import>
```

语法 2：

```
<c:import url="url" varReader="name" [context="context"]
[charEncoding="encoding"]>
   [标签体]
</c:import>
```

<c:import> 标签的属性说明如表 9.4 所示。

表 9.4　<c:import> 标签的属性说明

属　性	说　明
url	用于指定被导入的文件资源的 URL 地址。需要注意的是，如果指定的 url 属性为 null 或者无效，将抛出“javax.servlet.ServletException”异常。例如，如果指定 url 属性值为 url.jsp，而与当前页面同级的目录中并不存在 url.jsp 文件，将抛出“javax.servlet.ServletException: File "/url.jsp" not found”异常
context	上下文路径，用于访问同一个服务器的其他 Web 应用，其值必须以“/”开头，如果指定了该属性，那么 url 属性值也必须以“/”开头
var	用于指定变量名称。该变量用于以 String 类型存储获取的资源
scope	用于指定变量的存在范围，默认值为 page。可选值有 page、request、session 和 application
varReader	用于指定一个变量名，该变量用于以 Reader 类型存储被包含文件内容

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（续）

属　性	说　明
charEncoding	用于指定被导入文件的编码格式
标签体	可选，如果需要为导入的文件传递参数，则可以在标签体的位置通过 <c:param> 标签设置参数

注意 导出的 Reader 对象，只能在 <c:import> 标记的开始标签和结束标签之间使用。

【例 9.5】 应用 <c:import> 标签导入网站 Banner。

实例位置：光盘 \MR\ 源码 \ 第 9 章 \9.5

Step 1) 编写 index.jsp 文件，在该文件中，首先应用 taglib 指令引用 JSTL 的核心标签库，然后应用 <c:set> 标签将歌曲类别列表组成的字符串保存到变量 typeName 中，最后应用 <c:import> 标签导入网站 Banner（对应的文件为 navigation.jsp），并将歌曲类别列表组成的字符串传递到 navigation.jsp 页面。index.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>应用 &lt;c:import&gt; 标签导入网站 Banner</title>
<style>
li {       padding: 5px;}
</style>
</head>
<body>
<body style=" margin:0px;">
<c:set var="typeName" value="流行金曲 | 经典老歌 | 热舞 DJ | 欧美金曲 | 少儿歌曲 | 轻音乐 |
最新上榜 "/>
<!-- 导入网站的 Banner -->
<c:import url="navigation.jsp" charEncoding="UTF-8">
   <c:param name="typeList" value="${typeName}"/>
</c:import>
</body>
</html>
```

Step 2) 编写 navigation.jsp 文件，在该文件中，首先设置请求的编码方式为 UTF-8（中文），然后添加一个 <header> 标记，通过 CSS 样式控制 <header> 标记的样式，并在该标记中通过 EL 输出 <c:param> 标签传递的参数值，navigation.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%request.setCharacterEncoding("UTF-8"); %>
<style>
header{
   margin:0 auto auto auto;                          /* 设置居中显示 */
   width:891px;                                      /* 设置宽度 */
   height:30px;                                      /* 设置高度 */
   background-image:url("images/bg.jpg");            /* 设置背景图片 */
   padding-top:98px;                                 /* 设置顶内边距 */
   font-weight: bold;                                /* 设置文字加粗显示 */
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    color: white;                                    /* 设置文字颜色为白色 */
    padding-left:10px;                               /* 设置左内边距 */
}
</style>
<header>${param.typeList}</header>
```

运行本实例，将显示如图 9.16 所示的运行结果。

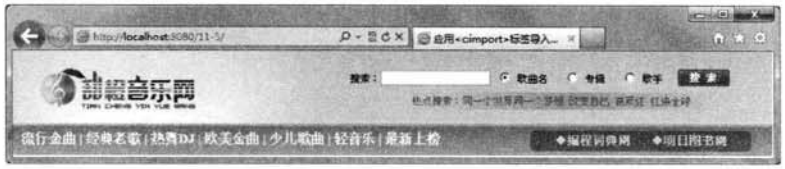


图 9.16　应用 <c:import> 标签导入网站 Banner

9.4.2　<c:url> 动态生成 URL 标签

<c:url> 标签用于生成一个 URL 路径的字符串，这个生成的字符串可以赋予 HTML 的 <a> 标记以实现 URL 的连接，或者用这个生成的 URL 字符串实现网页转发与重定向等。在使用该标签生成 URL 时，还可以与 <c:param> 标签相结合动态添加 URL 的参数信息。<c:url> 标签的语法格式如下。

语法 1：

```
<c:url value="url" [var="name"] [scope=" 范围 "] [context="context"]/>
```

该语法将输出产生的 URL 字符串信息，如果指定了 var 和 scope 属性，相应的 URL 信息就不再输出，而是存储在变量中以备后用。

语法 2：

```
<c:url value="url" var="name"] [scope=" 范围 "] [context="context"]>
   <c:param/>
   …       <!-- 可以有多个 <c:param> 标签 -->
</c:url>
```

该语法不仅实现了语法 1 的功能，而且还可以搭配 <c:param> 标签完成带参数的复杂 URL 信息。

- value 属性：用于指定将要处理的 URL 地址，可以使用 EL。
- context 属性：上下文路径，用于访问同一个服务器的其他 Web 工程，其值必须以“/”开头，如果指定了该属性，那么 url 属性值也必须以“/”开头。
- var 属性：用于指定变量名称，该变量用于保存新生成的 URL 字符串。
- scope 属性：用于指定变量的存在范围。

【例 9.6】　应用 <c:url> 标签生成带参数的 URL 地址。

实例位置：光盘 \MR\ 源码 \ 第 9 章 \9.6

编写 index.jsp 文件，在该文件中，首先应用 taglib 指令引用 JSTL 的核心标签库，然后应用 <c:url> 标签和 <c:param> 标签生成带参数的 URL 地址，并保存到变量 path 中，最后，添加一个超链接，该超链接的目标地址是 path 变量所指定的 URL 地址。index.jsp 文件的具体代码

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
… <!-- 此处省略了部分 HTML 代码 -->
<c:url var="path" value="register.jsp" scope="page">
   <c:param name="user" value="mr"/>
   <c:param name="email" value="wgh717@sohu.com"/>
</c:url>
<a href="${pageScope.path }"> 提交注册 </a>
</body>
</html>
```

运行本实例，将鼠标指针移动到“提交注册”超链接上，在状态栏中将显示生成的 URL 地址，如图 9.17 所示。

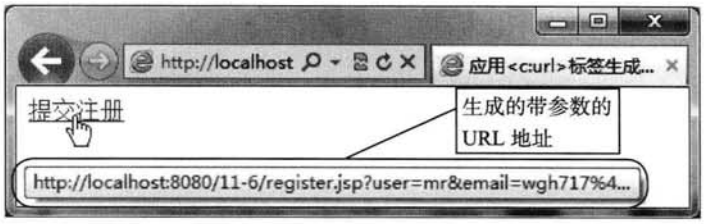


图 9.17　应用 <c:import> 标签导入网站 Banner

说明　在应用 <c:url> 标签生成新的 URL 地址时，空格符将被转换为加号“+”。

9.4.3　<c:redirect> 重定向标签

<c:redirect> 标签可以将客户端发出的 request 请求重定向到其他 URL 服务端，由其他程序处理客户的请求。而在这期间可以对 request 请求中的属性进行修改或添加，然后把所有属性传递到目标路径。该标签的语法格式如下。

语法 1：该语法格式没有标签体，并且不添加传递到目标路径的参数信息。

```
<c:redirect url="url" [context="/context"]/>
```

语法 2：该语法格式将客户请求重定向到目标路径，并且在标签体中使用 <c:param> 标签传递其他参数信息。

```
<c:redirect url="url" [context="/context"]>
   <c:param/>
… <!-- 可以有多个 <c:param> 标签 -->
</c:redirect>
```

- url 属性：必选属性，用于指定待定向资源的 URL，可以使用 EL。
- context 属性：在使用相对路径访问外部 context 资源时，用于指定资源的名字。

例如，应用语法 1 将页面重定向到用户登录页面的代码如下：

```
<c:redirect url="login.jsp"/>
```

再例如，应用语法 2 将页面重定向到 Servlet 映射地址 UserListServlet，并传递 action 参数，参数值为 query，具体代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<c:redirect url="UserListServlet">
   <c:param name="action" value="query"/>
</c:redirect>
```

9.4.4　<c:param> 传递参数标签

<c:param> 标签只用于为其他标签提供参数信息，它与 <c:import>、<c:redirect> 和 <c:url> 标签组合可以实现动态定制参数，从而使标签可以完成更复杂的程序应用。<c:param> 标签的语法格式如下：

```
<c:param name="paramName" value="paramValue"/>
```

- name 属性：用于指定参数名，可以引用 EL。如果参数名为 null 或是空，该标签将不起任何作用。
- value 属性：用于指定参数值，可以引用 EL。如果参数值为 null，将作为空值处理。

【例 9.7】 应用 <c:redirect> 和 <c:param> 标签实现重定向页面并传递参数。

实例位置：光盘 \MR\ 源码 \ 第 9 章 \9.7

Step 1) 编写 index.jsp 文件，在该文件中，首先应用 taglib 指令引用 JSTL 的核心标签库，然后应用 <c:redirect> 标签将页面重定向到 main.jsp 中，并且通过 <c:param> 标签传递用户名参数。index.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title> 重定向页面并传递参数 </title>
</head>
<body>
<c:redirect url="main.jsp">
   <c:param name="user" value="wgh"/>
</c:redirect>
</body>
</html>
```

Step 2) 编写 main.jsp 文件，在该文件中，通过 EL 显示传递的参数 user。main.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title> 显示结果 </title>
</head>
<body>
[${param.user }] 您好，欢迎访问我公司网站！
</body>
</html>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

运行本实例，将页面重定向到 main.jsp 页面，并显示传递的参数，如图 9.18 所示。

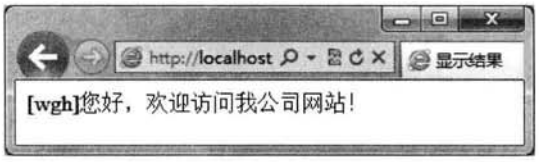


图 9.18　获取传递的参数

9.5　流程控制标签

通过流程控制功能，在程序中会根据不同的条件去执行不同的代码，可以产生不同的运行结果，从而可以处理程序中的任何可能发生的事情。在 JSTL 中包含 <c:if> 标签、<c:choose> 标签、<c:when> 标签和 <c:otherwise> 标签 4 种流程控制标签。

9.5.1　<c:if> 条件判断标签

<c:if> 条件判断标签可以根据不同的条件去处理不同的业务。它与 Java 语言中的 if 语句类似，只不过该语句没有 else 标签。<c:if> 标签有两种语法格式。

说明　虽然 <c:if> 标签没有对应的 else 标签，但是 JSTL 提供了 <c:choose>、<c:when> 和 <c:otherwise> 标签，可以实现 if else 的功能。

语法 1：该语法格式会判断条件表达式，并将条件的判断结果保存在 var 属性指定的变量中，而这个变量存在于 scope 属性所指定的范围中。

```
<c:if test="condition" var="name" [scope=page|request|session|application]/>
```

语法 2：该语法格式不但可以将 test 属性的判断结果保存在指定范围的变量中，还可以根据条件的判断结果去执行标签体。标签体可以是 JSP 页面能够使用的任何元素，例如，HTML 标记、Java 代码或者其他嵌入的 JSP 标签。

```
<c:if test="condition" var="name" [scope="范围"]>
   标签体
</c:if>
```

- test 属性：必选属性，用于指定条件表达式，可以使用 EL。
- var 属性：可选属性，用于指定变量名，该变量用于保存 test 属性的判断结果。如果该变量不存在就创建它。
- scope 属性：用于指定变量的有效范围，默认值为 page。可选值有 page、request、session 和 application。

【例 9.8】　应用 <c:if> 标签根据是否登录显示不同的内容。

实例位置：光盘 \MR\ 源码 \ 第 9 章 \9.8

编写 index.jsp 文件，在该文件中，首先应用 taglib 指令引用 JSTL 的核心标签库；然后应用 <c:if> 标签判断保存用户名的参数 username 是否为空，并将判断结果保存到变量 result 中，如果 username 为空，则显示用于输入用户信息的表单及表单元素；最后再判断变量 result 的值

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

是否为 true，如果不为 true，则通过 EL 输出当前登录用户及欢迎信息。index.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%request.setCharacterEncoding("UTF-8"); %>
… <!-- 此处省略了部分 HTML 代码 -->
<c:if var="result" test="${empty param.username}">
  <form name="form1" method="post" action="">
   用户名：<input name="username" type="text" id="username">
      <br><br>
      <input type="submit" name="Submit" value=" 登录 ">
  </form>
</c:if>
<c:if test="${!result}">
   [${param.username }] 欢迎访问我公司网站!
</c:if>
</body>
</html>
```

运行本实例，将显示如图 9.19 所示的页面，在“用户名”文本框输入用户名“无语”，单击“登录”按钮，将显示如图 9.20 所示的页面，显示欢迎信息。



图 9.19　未登录时显示的内容



图 9.20　登录后显示的内容

9.5.2　<c:choose> 条件选择标签

<c:choose> 标签可以根据不同的条件去完成指定的业务逻辑，如果没有符合的条件会执行默认条件的业务逻辑。<c:choose> 标签只能作为 <c:when> 和 <c:otherwise> 标签的父标签，而要实现条件选择逻辑可以在 <c:choose> 标签中嵌套 <c:when> 和 <c:otherwise> 标签来完成。<c:choose> 标签的语法格式如下：

```
<c:choose>
   标签体   <!-- 由 <c:when> 标签和 <c:otherwise> 标签组成 -->
</c:choose>
```

<c:choose> 标签没有相关属性，它只是作为 <c:when> 和 <c:otherwise> 标签的父标签来使用，并且在 <c:choose> 标签中，除了空白字符外，只能包括 <c:when> 和 <c:otherwise> 标签。

在一个 <c:choose> 标签中可以包含多个 <c:when> 标签来处理不同条件的业务逻辑，但是只能有一个 <c:otherwise> 标签来处理默认条件的业务逻辑。

说明　在运行时，首先判断 <c:when> 标签的条件是否为 true，如果为 true，则将 <c:when> 标签体中的内容显示到页面中，否则判断下一个 <c:when> 标签的条件，如果该标签的条件也不满足，则继续判断下一个 <c:when> 标签，直到 <c:otherwise> 标签体被执行。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【例 9.9】 应用 <c:choose> 标签根据是否登录显示不同的内容。

实例位置：光盘 \MR\ 源码 \ 第 9 章 \9.9

编写 index.jsp 文件，在该文件中，首先应用 taglib 指令引用 JSTL 的核心标签库；然后添加 <c:choose> 标签，在该标签中应用 <c:when> 标签判断保存用户名的参数 username 是否为空，如果 username 为空，则显示用于输入用户信息的表单及表单元素，否则应用 <c:otherwise> 标签处理不为空的情况，这里将通过 EL 输出当前登录用户及欢迎信息。index.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%request.setCharacterEncoding("UTF-8"); %>
… < !-- 此处省略了部分 HTML 代码 -->
<c:choose>
   <c:when test="${empty param.username}">
     <form name="form1" method="post" action="">
          用户名:
        <input name="username" type="text" id="username">
       <input type="submit" name="Submit" value=" 登录 ">
     </form>
   </c:when>
   <c:otherwise>
          [${param.username }] 欢迎访问我公司网站!
   </c:otherwise>
</c:choose>
</body>
</html>
```

运行本实例，将显示如图 9.21 所示的页面，在“用户名”文本框输入用户名 bellflower，单击“登录”按钮，将显示如图 9.22 所示的页面，显示欢迎信息。



图 9.21 未登录时显示的信息



图 9.22 登录后显示的内容

9.5.3 <c:when> 条件测试标签

<c:when> 条件测试标签是 <c:choose> 标签的子标签，它根据不同的条件去执行相应的业务逻辑，可以存在多个 <c:when> 标签来处理不同条件的业务逻辑。<c:when> 标签的语法格式如下：

```
<c:when test="condition">
   标签体
</c:when>
```

属性：test 为条件表达式，这是 <c:when> 标签必须定义的属性，它可以引用 EL 表达式。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明 在 <c:choose> 标签中，必须有一个 <c:when> 标签，但是 <c:otherwise> 标签是可选的。如果省略了 <c:otherwise> 标签，当所有的 <c:when> 标签都不满足条件时，将不会处理 <c:choose> 标签的标签体。

注意 <c:when> 标签必须出现在 <c:otherwise> 标签之前。

【例 9.10】 实现分时问候。

实例位置：光盘 \MR\ 源码 \ 第 9 章 \9.10

编写 index.jsp 文件，在该文件中，首先应用 taglib 指令引用 JSTL 的核心标签库；然后应用 <c:set> 标签定义两个变量，分别用于保存当前的小时数和分钟数；接下来再添加 <c:choose> 标签，在该标签中，应用 <c:when> 标签进行分时判断，并显示不同的问候信息；最后应用 EL 输出当前的小时和分钟数。index.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
… <!-- 此处省略了部分 HTML 代码 -->
<!-- 获取小时并保存到变量中 -->
<c:set var="hours">
   <%=new java.util.Date().getHours()%>
</c:set>
<!-- 获取分钟并保存到变量中 -->
<c:set var="second">
   <%=new java.util.Date().getMinutes()%>
</c:set>
<c:choose>
   <c:when test="${hours>1 && hours<7}">早上好！ </c:when>
   <c:when test="${hours>=7 && hours<12}" >上午好！ </c:when>
   <c:when test="${hours>=12 && hours<18}">下午好！ </c:when>
   <c:when test="${hours>=18 && hours<24}">晚上好！ </c:when>
</c:choose>
 现在时间是：${hours}:${second}
</body>
</html>
```

运行本实例，将显示如图 9.23 所示的问候信息。

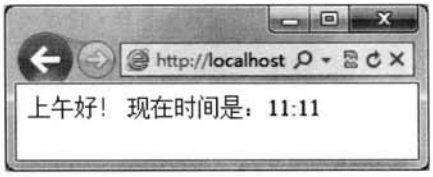


图 9.23　运行结果

9.5.4　<c:otherwise> 其他条件标签

<c:otherwise> 标签也是 <c:choose> 标签的子标签，用于定义 <c:choose> 标签中的默认条件处理逻辑，如果没有任何一个结果满足 <c:when> 标签指定的条件，将会执行这个标签体中定义的逻辑代码。在 <c:choose> 标签范围内只能存在一个该标签的定义。<c:otherwise> 标签的

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

语法格式如下：

```
<c:otherwise>
标签体
</c:otherwise>
```

注意 <c:otherwise> 标签必须定义在所有 <c:when> 标签的后面，也就是说它是 <c:choose> 标签的最后一个子标签。

【例 9.11】 幸运大抽奖。

实例位置：光盘 \MR\ 源码 \ 第 9 章 \9.11

编写 index.jsp 文件，在该文件中，首先应用 taglib 指令引用 JSTL 的核心标签库；然后抽取幸运数字并保存到变量中；最后再应用 <c:choose> 标签、<c:when> 标签和 <c:otherwise> 标签根据幸运数字显示不同的中奖信息。index.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ page import="java.util.Random" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
… <!-- 此处省略了部分 HTML 代码 -->
<%Random rnd=new Random();%>
<!-- 将抽取的幸运数字保存到变量中 -->
<c:set var="luck">
   <%=rnd.nextInt(10)%>
</c:set>
<c:choose>
   <c:when test="${luck==6}">恭喜你，中了一等奖！ </c:when>
   <c:when test="${luck==7}" >恭喜你，中了二等奖！ </c:when>
   <c:when test="${luck==8}">恭喜你，中了三等奖！ </c:when>
   <c:otherwise>谢谢您的参与！ </c:otherwise>
</c:choose>
</body>
</html>
```

运行本实例，当产生随机数 6 时，将显示如图 9.24 所示的中奖信息。

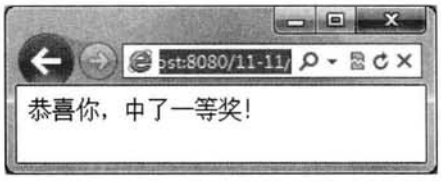


图 9.24 运行结果

9.6 循环标签

循环是程序算法中的重要环节，有很多著名的算法都需要在循环中完成，例如递归算法、查询算法和排序算法都需要在循环中完成。JSTL 标签库中包含 <c:forEach> 和 <c:forTokens> 两个循环标签。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

9.6.1　<c:forEach> 循环标签

<c:forEach> 循环标签可以根据循环条件，遍历数组和集合类中的所有或部分数据。例如，在使用 Hibernate 技术访问数据库时，返回的都是数组、java.util.List 和 java.util.Map 对象，它们封装了从数据库中查询出的数据，这些数据都是 JSP 页面需要的。如果在 JSP 页面中使用 Java 代码来循环遍历所有数据，会使页面非常混乱，不易分析和维护。使用 JSTL 的 <c:forEach> 标签循环显示这些数据不但可以解决 JSP 页面混乱的问题，而且也提高了代码的可维护性。

<c:forEach> 标签的语法格式如下。

语法 1：集合成员迭代

```
<c:forEach items="data" [var="name"] [begin="start"] [end="finish"] [step="step"]
[varStatus="statusName"]>
   标签体
</c:forEach>
```

在该语法中，items 属性是必选属性，通常使用 EL 指定，其他属性均为可选属性。

语法 2：数字索引迭代

```
<c:forEach begin="start" end="finish" [var="name"] [varStatus="statusName"]
[step="step"]>
   标签体
</c:forEach>
```

在该语法中，各属性的说明如表 9.5 所示，在这些属性中，begin 和 end 属性是必选的属性，其他属性均为可选属性。

表 9.5　<c:forEach> 标签的常用属性

属　性	说　明
items	用于指定被循环遍历的对象，多用于数组与集合类。该属性的属性值可以是数组、集合类、字符串和枚举类型，并且可以通过 EL 进行指定
var	用于指定循环体的变量名，该变量用于存储 items 指定的对象的成员
begin	用于指定循环的起始位置，如果没有指定，则从集合的第一个值开始迭代，可以使用 EL
end	用于指定循环的终止位置，如果没有指定，则一直迭代到集合的最后一位，可以使用 EL
step	用于指定循环的步长，可以使用 EL
varStatus	用于指定循环的状态变量，该属性还有 4 个状态属性，如表 9.6 所示
标签体	可以是 JSP 页面可以显示的任何元素

表 9.6　状态属性

变　量	类　型	描　述
index	int	当前循环的索引值，从 0 开始
count	int	当前循环的循环计数，从 1 开始
first	boolean	是否为第一次循环
last	boolean	是否为最后一次循环

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明 在循环的过程中，如果要得到循环计数，可以应用 varStatus 属性的状态属性 count 获得。

【例 9.12】 遍历 List 集合。

实例位置：光盘 \MR\ 源码 \ 第 9 章 \9.12

编写 index.jsp 文件，在该文件中，首先应用 taglib 指令引用 JSTL 的核心标签库；然后创建一个包含 3 元素的 List 集合对象，并保存到 request 范围内的 list 变量中；接下来应用 <c:forEach> 标签遍历 List 集合中的全部元素并输出；最后应用 <c:forEach> 标签遍历 List 集合中第 1 个元素以后的元素并输出，不包括第 1 个元素。index.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ page import="java.util.*" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
… <!-- 此处省略了部分 HTML 代码 -->
<%
List<String> list=new ArrayList<String>();          // 创建 List 集合的对象
list.add(" 简单是可靠的先决条件 ");                    // 向 List 集合中添加元素
list.add(" 兴趣是最好的老师 ");
list.add(" 知识上的投资总能得到最好的回报 ");
request.setAttribute("list",list);                   // 将 List 集合保存到 request 对象中
%>
<b> 遍历 List 集合的全部元素：</b><br>
<c:forEach items="${requestScope.list}" var="keyword" varStatus="id">
   ${id.index } : ${keyword}<br>
</c:forEach>
<b> 遍历 List 集合中第 1 个元素以后的元素（不包括第 1 个元素）：</b><br>
<c:forEach items="${requestScope.list}" var="keyword" varStatus="id" begin="1">
   ${id.index } : ${keyword}<br>
</c:forEach>
</body>
</html>
```

说明 在应用 <c:forEach> 标签时，var 属性指定的变量只在循环体内有效，这一点与 Java 语言的 for 循环语句中的循环变量类似。

运行本实例，将显示如图 9.25 所示的效果。

【例 9.13】 应用 <c:forEach> 列举 10 以内全部奇数。

实例位置：光盘 \MR\ 源码 \ 第 9 章 \9.13

编写 index.jsp 文件，在该文件中，首先应用 taglib 指令引用 JSTL 的核心标签库，然后应用 <c:forEach> 标签输出 10 以内全部奇数。index.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
… <!-- 此处省略了部分 HTML 代码 -->
<b>10 以内的全部奇数为：</b>
<c:forEach var="i" begin="1" end="10" step="2">
   ${i}  
</c:forEach>
</body>
</html>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

运行本实例，将显示如图 9.26 所示的效果。

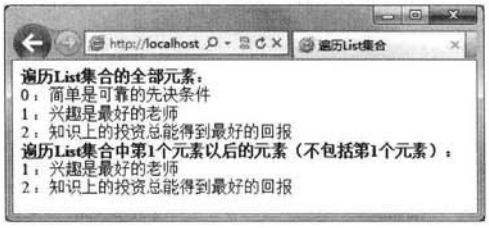


图 9.25　运行结果图

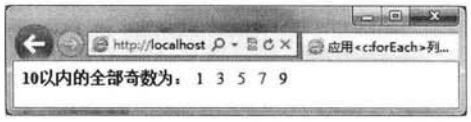


图 9.26　列举 10 以内全部奇数

9.6.2　<c:forTokens> 迭代标签

<c:forTokens> 迭代标签可以用指定的分隔符将一个字符串分割开，根据分割的数量确定循环的次数。<c:forTokens> 标签的语法格式如下：

```
<c:forTokens items="String" delims="char" [var="name"] [begin="start"] [end="end"]
[step="len"] [varStatus="statusName"]>
   标签体
</c:forTokens>
```

<c:forTokens> 标签的常用属性如表 9.7 所示。

表 9.7　<c:forTokens> 标签的常用属性

属　性	说　明
items	用于指定要迭代的 String 对象，该字符串通常由指定的分隔符分隔
delims	用于指定分隔字符串的分隔符，可以同时有多个分隔符
var	用于指定变量名，该变量中保存了分隔后的字符串
begin	用于指定迭代开始的位置，默认索引值从 0 开始
end	用于指定迭代的结束位置
step	用于指定迭代的步长，默认步长为 1
varStatus	用于指定循环的状态变量，同 <c:forEach> 标签一样，该属性也有 4 个状态属性，如表 9.6 所示
标签体	可以是 JSP 页面可以显示的任何元素

【例 9.14】 应用 <c:forTokens> 分隔字符串。

实例位置：光盘 \MR\ 源码 \ 第 9 章 \9.14

编写 index.jsp 文件，在该文件中，首先应用 taglib 指令引用 JSTL 的核心标签库；然后应用 <c:set> 标签定义一个字符串变量，并输出该字符串；最后应用 <c:forTokens> 标签迭代输出按指定分隔符分隔的字符串。index.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
… <!-- 此处省略了部分 HTML 代码 -->
<c:set var="sourceStr" value="Java Web：程序开发范例宝典、典型模块大全；Java：实例完全自
学手册、典型模块大全 "/>
<b> 原字符串：</b><c:out value="${sourceStr}"/>
<br><b> 分割后的字符串：</b><br>
<c:forTokens items="${sourceStr}" delims="：、；" var="item">
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    ${item}<br>
</c:forTokens>
</body>
</html>
```

运行本实例，将显示如图 9.27 所示的效果。

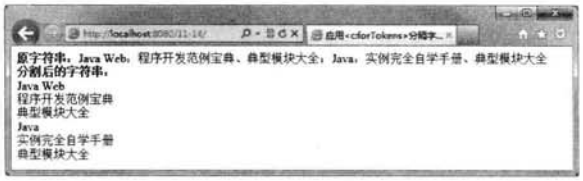


图 9.27　运行结果

9.7　上机实践

9.7.1　利用 JSTL 标签导入用户注册协议

➤➤➤ 题目描述

在网站的使用过程中，用户注册之前需要显示用户注册协议信息，只有同意协议才可以继续完成注册。本上机实践尝试应用 JSTL 标签中的 <c:import> 标签导入注册协议的文本资源文件，运行结果如图 9.28 所示。

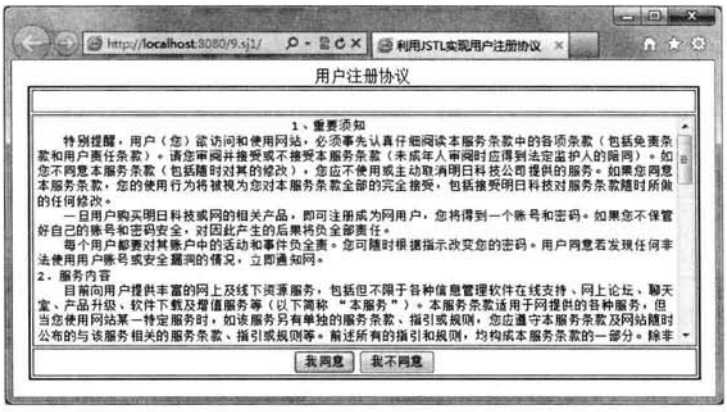


图 9.28　利用 JSTL 标签导入用户注册协议

➤➤➤ 技术指导

本实例主要应用 JSTL 标签库中的 <c:import> 标签来实现。<c:import> 标签用于导入其他 Web 资源，与 <jsp:include> 的作用类似。区别在于它不仅可以导入同一个 Web 应用中的资源，还可以导入其他 Web 应用中的资源，甚至是其他网站的资源。

本上机实践的关键代码参考如下：

```
<c:import url="agreement.txt" charEncoding="UTF-8"/>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

9.7.2　根据参数请求显示不同的页面

➤➤➤ 题目描述

通过 JSTL 标签库中的 <c:if> 标签可以根据不同的条件去处理不同的业务。本上机实践尝试应用 <c:if> 标签实现根据参数请求显示不同页面的功能。运行结果如图 9.29 所示。在页面中将根据 <c:if> 标签判断并显示不同提示信息。

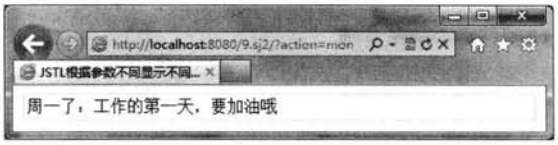


图 9.29　根据参数请求显示不同的页面

➤➤➤ 技术指导

本实例主要应用 JSTL 标签库中的 <c:if> 标签来实现。<c:if> 标签能够实现 Java 语言中的 if 语句及 if-else 语句的功能。

本上机实践的关键代码参考如下：

```
<c:if test="${ param.action == 'mon' }">
                周一了：工作的第一天，要加油哦
</c:if>
<c:if test="${ param.action == 'tues' }">
                周二了：工作了两天了，要适当补充体力哦
</c:if>
<c:if test="${ param.action == 'wed' }">
                周三了：忙碌的生活要学会调节
</c:if>
<c:if test="${ param.action == 'thu' }">
                周四了：偶尔偷下懒儿，不算过分哦
</c:if>
<c:if test="${ param.action == 'fri' }">
        周五了：加油，明天就要休息了，HOHO
</c:if>
<c:if test="${ param.action == 'sat' }">
        周六了：和死党们出去 HAPPY 吧
</c:if>
<c:if test="${ param.action == 'sun' }">
        周日：要收敛一下活动，明天要上班呢
</c:if>
```

9.7.3　JSTL 在电子商城中的应用

➤➤➤ 题目描述

在实现电子商城网站时，通常要实现用户登录、显示分时问候和重定向页面等功能，这可以通过使用 JSTL 来实现。本上机实践尝试应用 JSTL 实现电子商城网站中的用户登录、显示分时问候和页面重定向等功能，运行结果如图 9.30 和图 9.31 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 9.30　未登录时的页面运行结果



图 9.31　登录后的页面运行结果

➤➤➤ 技术指导

本实例主要应用 JSTL 标签库中的 <c:choose>、<c:when>、<c:otherwise>、<c:redirect>、<c:import> 标签来实现。其中，<c:choose>、<c:when> 和 <c:otherwise> 用于实现显示分时问候和根据用户是否登录显示不同的内容；<c:redirect> 用于重定向页面；<c:import> 标签用于包含页头、登录页面和版权页面。

本上机实践的显示分时问候的关键代码参考如下：

```
<div id="greeting">
   <jsp:useBean id="now" class="java.util.Date"/>
   <c:choose>
          <c:when test="${now.hours>=0 && now.hours<5}">
                 凌晨好!
          </c:when>
          <c:when test="${now.hours>=5 && now.hours<8}">
                 早上好!
          </c:when>
          <c:when test="${now.hours>=8 && now.hours<11}">
                 上午好!
          </c:when>
          <c:when test="${now.hours>=11 && now.hours<13}">
                 中午好!
          </c:when>
          <c:when test="${now.hours>=13 && now.hours<17}">
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                下午好!
            </c:when>
            <c:otherwise>
            晚上好!
            </c:otherwise>
        </c:choose>
        现在时间是：${now.hours}时${now.minutes}分${now.seconds}秒
    </div>
```

9.8　小结

本章首先简要介绍了JSTL标签库，然后详细介绍了JSTL标签库的下载和配置，以及JSTL核心标签库中的表达式标签、URL相关标签、流程控制标签和循环标签。其中，JSTL的核心标签库在实际项目开发中比较常用，需要读者重点掌握。

9.9　习题

选择题

1. 下面哪个标签库的功能与if语句类似？（　　）

 A. <c:out>　　B. <c:if>　　C. <c:set>

2. <c:set>标签的功能为（　　）。

 A. 在页面输出信息　　B. 设置变量属性值　　C. 获取异常信息

3. 要将文件导入到JSP页面，可以使用（　　）标签。

 A. <c:import>　　B. <c:out>　　C. <c:url>

4. 要在页面中输出信息，使用的是（　　）标签。

 A. <c:import>　　B. <c:out>　　C. <c:url>

5. 下面哪个标签可用于循环？（　　）

 A. <c:import>　　B. <c:out>　　C. <c:forEach>

填空题

1. <c:out>标签中的value属性的含义为_____。
2. JSTL提供了_____5种标签库。
3. _____几种标签可实现流程控制。
4. _____几种标签可实现循环控制。
5. _____标签可实现页面重定向。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 10 章　数据库应用开发

视频讲解：61 分钟

数据库应用在日常的生活和工作中可以说是无处不在，无论是一个小型的企业办公自动化系统，还是像中国移动那样的大型运营系统，似乎都离不开数据库。对于大多数应用程序来说，不管它们是 Windows 桌面应用程序，还是 Web 应用程序，存储和检索数据都是其核心功能，所以针对数据库的开发已经成为软件开发的一种必备技能。如果说过去是“学好数理化，走遍天下都不怕”，那么，对于今天的软件开发者而言就是“学好数据库，走到哪儿都不怵”。本章将向读者介绍如何在 Java Web 中进行数据库应用开发。

通过学习本章，读者可以达到以下学习目的：

- 了解 Web 开发时常用的数据库；
- 了解 JDBC 及连接数据库的开发流程；
- 掌握 JDBC 核心 API；
- 掌握如何使用 JDBC 操作数据库；
- 掌握在 Java Web 开发时如何实现分页查询。

10.1　Web开发时常用的数据库

Java Web 可以访问并操作很多种数据库管理系统，如 SQL Server、MySQL、Oracle、Access、DB2、Sybase 和 PostgreSQL 等，下面介绍几种常用的数据库管理系统。

10.1.1　MySQL 数据库

MySQL 是目前最为流行的开放源码的数据库，是完全网络化的跨平台的关系型数据库系统，它是由 MySQL AB 公司开发、发布并支持的。任何人都能从 Internet 下载 MySQL 软件，而无须支付任何费用，并且“开放源码”意味着任何人都可以使用和修改该软件，如果愿意，用户也可以研究源码并进行恰当的修改，以满足自己的需求，不过需要注意的是，这种“自由”是有范围的。下面将介绍如何下载和安装 MySQL 数据库，以及如何使用 MySQL 的图形工具。

1．下载和安装 MySQL 数据库

MySQL 可以到 MySQL 的官方网站（http://www.mysql.com）中下载。目前最新的版本是 MySQL 5.5.24，下载后将得到名称为 mysql-installer-5.5.24.0.msi 的安装包文件。

MySQL 安装包下载完毕后，就可以通过该文件安装 MySQL 数据库了，具体的安装过程如下。

Step 1）双击下载后的 mysql-installer-5.5.24.0.msi 文件，打开安装向导对话框，如果没有打开安装向导对话框，而是弹出如图 10.1 所示的对话框，那么还需要先安装 .NET 4.0 框

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

架，再重新双击下载后的安装文件，打开安装向导对话框。

Step 2 在打开的安装向导对话框中，单击 Install MySQL Products 超链接，将打开 License Agreement 对话框，询问是否接受协议，选中 I accept the license terms 复选框，接受协议后，单击 Next 按钮，将打开 Find latest products 对话框。在该对话框中，选中 Skip the check for updates(not recommended) 复选框，这时，原来的 Execute 按钮将转换为 Next 按钮，如图 10.2 所示。



图 10.1　打开需要安装 .NET 4.0 框架的提示对话框

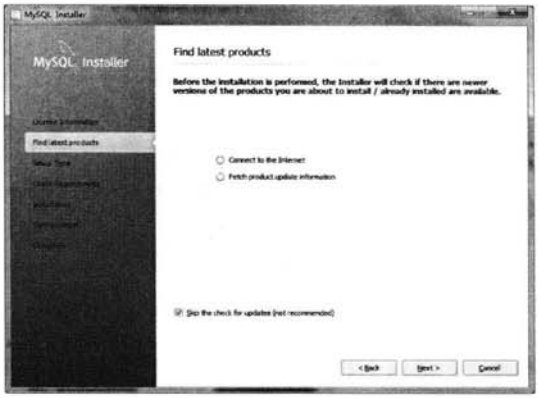


图 10.2　Find latest products 对话框

Step 3 单击 Next 按钮，将打开 Choosing a Setup Type 对话框，在该对话框中，共包括 Developer Default（开发者默认）、Server Only（仅服务器）、Client only（仅客户端）、Full（完全）和 Custom（自定义）5 种安装类型，这里选择开发者默认，并且将安装路径修改为“K:\Program Files\MySQL\”，数据存放路径修改为“K:\ProgramData\MySQL\MySQL Server 5.5”，如图 10.3 所示。

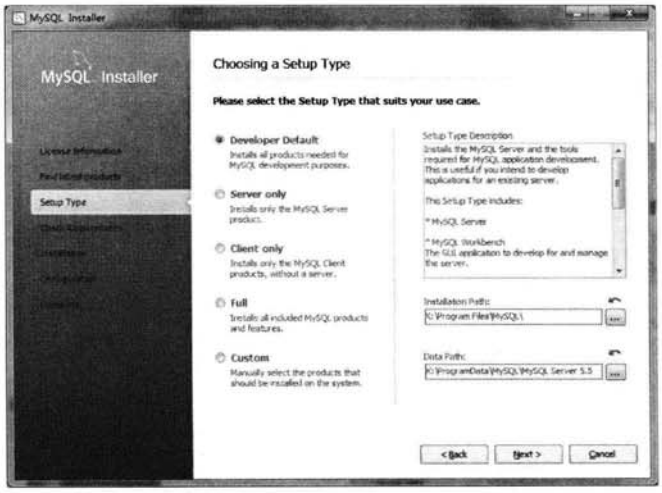


图 10.3　Choosing a Setup Type 对话框

Step 4 单击 Next 按钮，将打开 Check Requirements 对话框，在该对话框中检查系统是否具备安装所必需的 .NET 4.0 框架和 Microsoft Visual C++ 2010 32-bit runtime，如果不存在，单击

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Execute 按钮，将在线安装所需插件，安装完成后，将显示如图 10.4 所示的对话框。

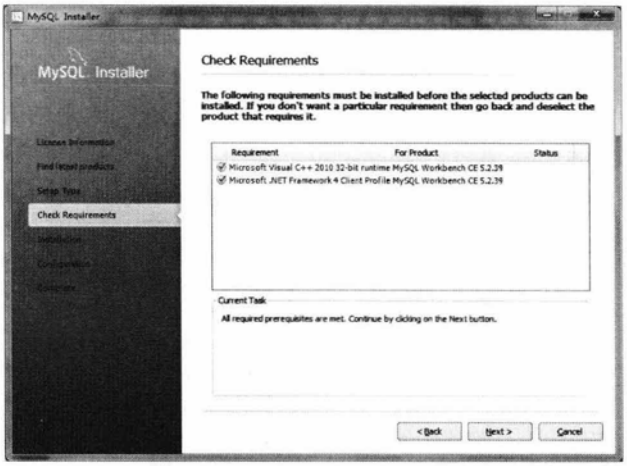


图 10.4　安装条件已全部满足时的 Check Requirements 对话框

Step 5 单击 Next 按钮，将打开 Installation Progress 对话框，单击 Execute 按钮，将开始安装，并显示安装进度。安装完成后，单击 Next 按钮，将打开 Configuration Overview 对话框，在该对话框中，单击 Next 按钮，将打开用于选择服务器类型的 MySQL Server Configuration 对话框，在该对话框中共提供了开发者类型、服务器类型和致力于 MySQL 服务类型。这里选择默认的开发者类型。单击 Next 按钮，将打开用于设置网络选项和安全的 MySQL Server Configuration 对话框，在这个对话框中，设置 root 用户的登录密码为 111，其他采用默认，如图 10.5 所示。

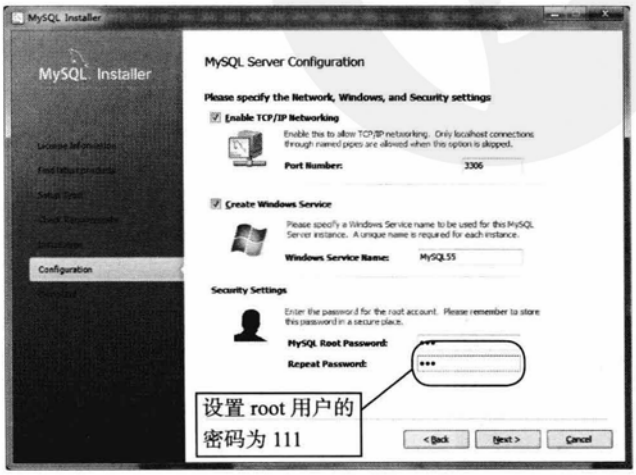


图 10.5　设置网络选项和安全的 MySQL Server Configuration 对话框

说明　MySQL 使用的默认端口是 3306，在安装时，可以修改为其他的端口，例如 3307。但是一般情况下，不要修改默认的端口号，除非 3306 端口已经被占用。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Step 6 单击 Next 按钮，将打开 Configuration Overview 对话框，开始配置 MySQL 服务器，配置完成后，单击 Next 按钮，继续配置，直到全部配置完成，然后单击 Finish 按钮，完成 MySQL 的安装。

2．使用 MySQL 的图形化工具

MySQL 数据库安装完成后，将自动安装了一个图形化工具，用于创建并管理数据库。在开始菜单中选择“所有程序”/MySQL/MySQL Workbench 5.2 CE 菜单项，将打开如图 10.6 所示的 MySQL 工作台界面。



图 10.6　MySQL 工作台界面

（1）打开 SQL Editor

在 MySQL 工作台界面中，双击 Local instance MySQL 55 超链接，将打开一个输入用户密码的对话框，在该对话框中输入 root 用户的密码，这里为 111，单击 OK 按钮，将打开如图 10.7 所示的 SQL Editor 选项卡。在 SQL Editor 选项卡中，可以创建 / 管理数据库、创建 / 管理数据表、编辑表数据和查询表数据等操作。



图 10.7　SQL Editor 选项卡

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（2）导入 / 导出数据

在 MySQL 工作台界面中，单击 Manage Import / Export 超链接，将打开 Admin 选项卡，在该选项卡中，可以导出或者导入数据。

❑ 导出数据。

单击左侧的 Data Export 列表项，在右侧将显示用于进行数据导出的相关内容。例如，我们要为数据库 db_test 导出对应的 SQL 脚本文件，可以进行以下操作。

首先在数据库列表中选择 db_test 数据库，然后在数据库列表的右侧将显示对应的数据表（可以选择要导出的数据表），接下来在下方的 Options 区域中选择 Export to Self-Contained File 单选按钮，并指定生成的脚本文件保存的位置及文件名，如图 10.8 所示，最后单击 Start Export 按钮，就可以将该数据库导出为 SQL 脚本了。

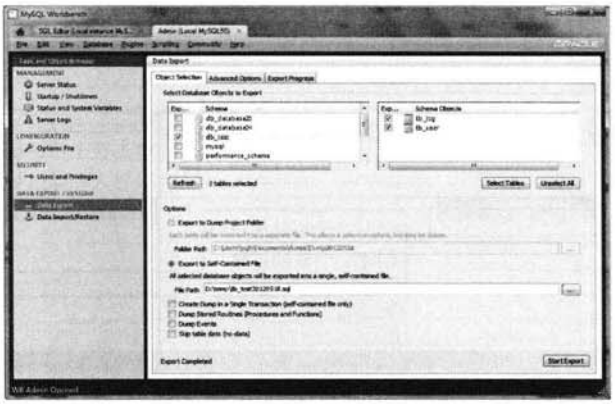


图 10.8　导出数据

❑ 导入数据。

单击左侧的 Data Import/Restore 列表项，在右侧将显示用于进行数据导入的相关内容。例如，我们要将一个数据库从 SQL 脚本文件还原回来，可以进行以下操作。

首先在 Options 区域中选择 Import from Self-Contained File 单选按钮，并单击“...”按钮选择要使用的 SQL 脚本文件，如图 10.9 所示，然后单击 Start Import 按钮，就可以将该数据库还原回来。

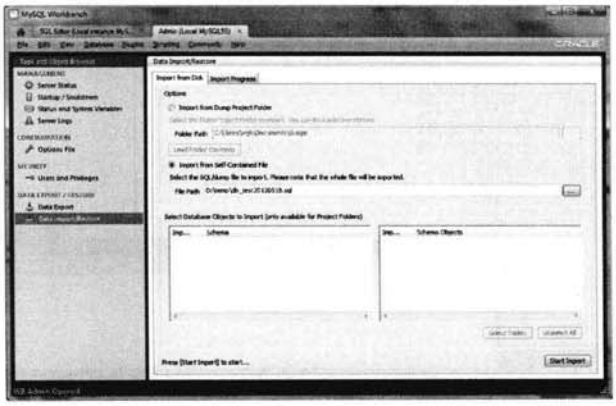


图 10.9　导入数据

Java学习群：72030155

好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

10.1.2　SQL Server 2008

SQL Server 2008 是一个重大的产品版本，它推出了许多新的特性和关键的改进，使得它成为至今为止最强大和最全面的 SQL Server 版本。SQL Server 是使用客户机 / 服务器体系结构的关系型数据库管理系统（RDBMS）。1996 年，微软公司推出了 SQL Server 6.5 版本。1998 年，推出了 SQL Server 7.0 版本。2000 年，推出了 SQL Server 2000。2005 年，推出了 SQL Server 2005，2008 年推出了 SQL Server 2008。目前，SQL Server 已经是世界上应用最普遍的大型数据库之一。

SQL Server 2008 不仅可以随时随地管理数据，而且可以将数据结构化、半结构化和非结构化文档的数据直接存储到数据库中。SQL Server 2008 提供了一系列丰富的集成服务，可以对数据进行查询、搜索、同步、报告和分析等操作。

10.1.3　Oracle 数据库

Oracle 系统是由以 RDBMS 为核心的一批软件产品组成，可在多种硬件平台上运行，例如微机、工作站、小型机、中型机和大型机等，并且支持多种操作系统，用户的 Oracle 应用可以方便地从一种计算机配置移至另一种计算机配置上。Oracle 的分布式结构可将数据和应用驻留在多台计算机上，并且相互间的通信是透明的。Oracle 支持大数据库、多用户的高性能的事务处理，支持大量用户同时在同一数据上执行各种数据应用，并使数据争用最小，保证数据一致性。系统维护具有很高的性能，甚至可 24 小时连续工作，正常的系统操作（非计算机系统故障）不会中断数据库的使用。

10.2　JDBC技术

JDBC 是 Java 程序与数据库系统通信的标准 API（Application Programming Interface，应用程序编程接口），定义在 JDK 的 API 之中。通过 JDBC 技术，Java 程序可以非常方便地与多种数据库交互。JDBC 在 Java 程序与数据库系统之间建立了一座桥梁。

10.2.1　JDBC 简介

JDBC 是 JDK 的重要组成部分之一，它提供了对多种类型数据库的统一访问方式。

不同种类的数据库（如 SQL Server、Oracle 和 MySQL 等）在其内部处理数据的方式不同，如果直接使用数据库厂商提供的访问接口操作数据库，应用程序的可移植性就会大打折扣。例如，用户当前在程序中使用 SQL Server 提供的接口操作数据库，如果采用 Oracle 数据库，则需要重新使用 Oracle 数据库提供的接口，代码的改动量巨大。有了 JDBC，这样的情况就不复存在了。它要求各个数据库厂商按照统一的规范来提供数据库驱动，程序中由 JDBC 和具体的数据库驱动联系。这样用户就不必直接与底层的数据库交互，代码的通用性更强。图 10.10 所示为 Java 程序通过 JDBC 访问数据库的方式。

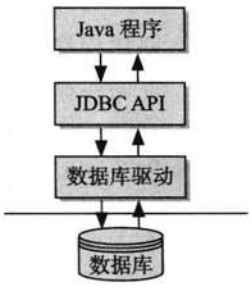


图 10.10　Java 程序通过 JDBC 访问数据库的方式

可以看出 JDBC 在 Java 程序与数据库之间起到了一个桥梁的作用，通过 JDBC 即可方便地与多种数据库交互，不必为某一个特

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

定的数据库编写专门的访问程序。例如，访问 MySQL 和 SQL Server 均可使用 JDBC。因此 JDBC 对 Java 开发人员而言，是一套标准的操作数据库的 API；对数据库厂商而言，又是一套标准的模型接口。

10.2.2 JDBC 连接数据库的开发流程

使用 JDBC 连接数据库的开发流程的关键步骤如下。

Step 1) 注册数据库驱动。

连接数据库前需要将数据库厂商提供的数据库驱动类注册到 JDBC 的驱动管理器中，通常情况下是将数据库驱动类加载到 JVM（Java Virtral Machine，Java 虚拟机），其代码如下：

```
Class.forName("com.mysql.jdbc.Driver");          // 加载数据库驱动，注册到驱动管理器
```

Step 2) 构建数据库连接的 URL。

建立数据库连接要构建数据库连接的 URL，这个 URL 由数据库厂商制定。不同数据库的 URL 不同，但基本格式均为“JDBC 协议 +IP 地址或域名 + 端口 + 数据库名称”。如 MySQL 的数据库连接 URL 的字符串为“jdbc:mysql://localhost:3306/test”。

Step 3) 获取 Connection 对象。

在注册数据库驱动及构建数据库 URL 后，即可通过驱动管理器获取数据库的 Connection 对象，该对象是 JDBC 封装的数据库连接对象。只有创建此对象后，才可以对数据执行相关操作，其获取方法如下：

```
DriverManager.getConnection(url,username,password);
```

获取 Connection 对象需要使用 DriverManager 对象，该对象的 getConnection() 方法通过数据库连接 URL、数据库用户名及数据库密码创建 Connection 对象。

【例 10.1】 通过 JDBC 连接 MySQL 数据库。

实例位置：光盘 \MR\ 源码 \ 第 10 章 \10.1

Step 1) 创建动态 Web 项目，将 MySQL 数据库的驱动包添加至项目的构建路径，以构建开发环境。

说明 在 JDK 中不包含数据库的驱动程序，使用 JDBC 操作数据库需要事先下载数据库厂商提供的驱动包。本实例使用 MySQL 5.1 数据库，所以添加的是 MySQL 官方提供的数据库驱动包，其名为 mysql-connector-java-5.1.20-bin.jar。

Step 2) 创建程序中的主页 index.jsp，在该页面中，首先通过 Class 的 forName() 方法加载数据库驱动，然后使用 DriverManager 对象的 getConnection() 方法获取数据库连接 Connection 对象，并将获取结果输出到页面中。index.jsp 页面的关键代码如下：

```
<%@ page import="java.sql.*"%>
<%
   try {
          Class.forName("com.mysql.jdbc.Driver");
                                          // 加载数据库驱动，注册到驱动管理器
          String url = "jdbc:mysql://localhost:3306/test";     // 数据库连接字符串
          String username = "root";                            // 数据库用户名
          String password = "111";                             // 数据库密码
          Connection conn = DriverManager.getConnection(url,username,password);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            // 创建 Connection 连接
            // 判断数据库连接是否为空
            if(conn != null){
                out.println("数据库连接成功！");                    // 输出连接信息
                conn.close();                                      // 关闭数据库连接
            }else{
                out.println("数据库连接失败！");                    // 输出连接信息
            }
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    %>
```

说明　Class 的 forName() 方法的作用是将指定字符串名的类加载到 JVM 中，实例中调用此方法来加载数据库驱动，加载后数据库驱动程序将会把驱动类自动注册到驱动管理器之中。

本实例的运行结果如图 10.11 所示。

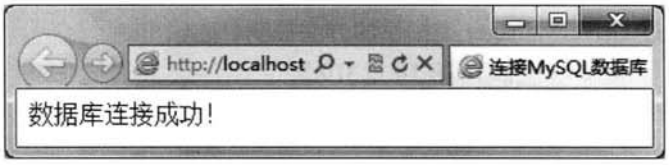


图 10.11　连接 MySQL 数据库

注意　只有数据库的服务处于开启状态，才能成功地与数据库建立连接。如果数据库连接失败，则应确认数据库服务是否开启。

10.3　JDBC核心API

JDBC 核心 API 主要包括 5 个接口，即代表数据库连接的 Connection 接口、驱动程序管理类 DriverManager、执行 SQL 语句的 Statement 接口、预编译 SQL 语句的 PreparedStatement 接口和结果集 ResultSet 接口。

10.3.1　驱动程序管理器 DriverManager

DriverManager 类负责管理 JDBC 驱动程序的基本服务，是 JDBC 的管理层。作用于用户和驱动程序之间，负责跟踪可用的驱动程序，并在数据库和驱动程序之间建立连接；另外，DriverManager 类也处理诸如驱动程序登录时间限制，以及显示登录和跟踪消息等。成功加载 Driver 类并在 DriverManager 类中注册后，DriverManager 类即可用来建立数据库连接。

当调用 DriverManager 类的 getConnection() 方法请求建立数据库连接时，该类将试图定位一个适当的 Driver 类，并检查定位到的 Driver 类是否可以建立连接。如果可以，则建立连接并返回；否则抛出 SQLException 异常。

DriverManager 类提供的常用方法如表 10.1 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 10.1 DriverManager 类提供的常用方法

方 法	说 明
deregisterDriver(Driver driver)	从 DriverManager 的管理列表中删除一个驱动程序，参数 driver 为要删除的驱动对象
getConnection(String url)	根据指定数据库连接 URL，建立与数据库的连接，参数 url 为数据库连接 URL
getConnection(String url,Properties info)	根据指定数据库连接 URL 及数据库连接属性信息建立数据库连接，参数 url 为数据库连接 URL，参数 inof 为数据库连接属性
getConnection(Stringurl,String user,Stringpassword)	根据指定数据库连接 URL、用户名及密码建立数据库连接，参数 url 为数据库连接 URL，参数 user 为连接数据库的用户名，参数 password 连接数据库的密码
getDrivers()	获取当前 DriverManager 中已加载的所有驱动程序，返回值为 Enumeration
registerDriver(Driver driver)	向 DriverManager 注册一个驱动对象，参数 driver 为要注册的驱动

【例 10.2】 通过 DriverManager 类提供的方法控制日志输出。

实例位置：光盘 \MR\ 源码 \ 第 10 章 \10.2

生成的日志文件如图 10.12 所示。

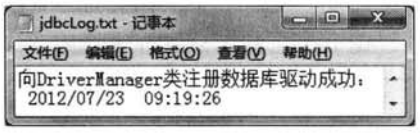


图 10.12 生成的日志文件

加载数据库驱动的代码如下：

```
String driverClass = "com.microsoft.jdbc.
sqlserver.SQLServerDriver";
Class.forName(driverClass);
```

获取 Web 工程的绝对路径的代码如下：

```
String url=request.getRealPath("");
```

本示例将日志信息输出到 Web 工程的 /WEB-INF/logs 路径下名为“jdbcLog.txt”的文件中。为此需要创建该文件的对象，代码如下：

```
File file = new File(url+"/WEB-INF/logs/jdbcLog.txt");
```

为上面的文件对象创建一个输出对象，代码如下：

```
PrintWriter printWriter = new PrintWriter(file);
```

通过 DriverManager 类的静态方法 setLogWriter() 将上面的输出对象注册到 DriverManager 类，这样该类产生的日志信息将输出到 Web 工程的 /WEB-INF/logs 路径下的名为“jdbcLog.txt”的文件中，代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
DriverManager.setLogWriter(printWriter);
```

下面通过 DriverManager 类的静态方法 println() 向日志文件输出追加的日志信息，代码如下：

```
DriverManager.println("向 DriverManager 类注册数据库驱动成功："+date);
```

10.3.2　代表数据库连接的 Connection 接口

Connection 接口用于创建数据库的连接（会话），只有获得该连接对象后才能访问数据库，并操作数据表。Connection 接口的常用方法如表 10.2 所示。

表 10.2　Connection 接口的常用方法

方　法	说　明
clearWarnings()	清除为此 Connection 对象报告的所有警告
close()	关闭连接，释放系统资源
commit()	使自从上一次提交 / 回滚以来执行的所有更改成为持久更改，并释放此 Connection 对象当前保存的所有数据库锁定
createStatement()	创建一个 Statement 对象
isClosed()	检索此 Connection 对象是否已经被关闭
prepareStatement()	创建一个 PreparedStatement 对象来将参数化的 SQL 语句发送到数据库
rollback()	取消在当前事务中进行的所有更改，并释放此 Connection 对象当前保存的所有数据库锁定

说明　表 10.2 中列出的是 Connection 接口的常用方法，如果要了解其他方法，请参考 Java API 或相关资料。

Connection 接口创建与数据库连接的目的是与数据库通信，为此需要执行 SQL 语句。但是通过 Connection 实例并不能执行 SQL 语句，还需要通过 Connection 实例创建 Statement 实例。

例如下面的代码，将通过 Connection 实例创建不同的 Statement 实例。

```
Connection conn=DriverManager.getConnection(url, username, password);  // 建立连接
//创建执行静态 SQL 语句的 Statement 实例
Statement stmt=conn.createStatement();                                  // 创建连接状态
//创建执行动态 SQL 语句的 Statement 实例（PrepareStatement 实例）
PreparedStatement ppdstmt = connection
        .prepareStatement("select * from table where id=?");
ppdstmt.setInt(1, 6);
ResultSet rs = ppdstmt.executeQuery();
//创建执行数据库存储过程的 Statement 实例（CallableStatement 实例）
CallableStatement cabstmt = connection.prepareCall("{call test_insert(?,?)}");
cabstmt.setInt(1, 6);
cabstmt.setString(2, "INSERT");
int c = cabstmt.executeUpdate();
```

10.3.3　执行 SQL 语句的 Statement 接口

Statement 接口用于执行静态的 SQL 语句，并返回一个生成结果的对象，该接口的对象通过 Connection 实例的 createStatement() 方法获得。利用该对象把静态 SQL 语句发送到数据库编译执行，然后返回数据库的处理结果。该接口的常用方法如表 10.3 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 10.3 Statement 接口的常用方法

方 法	说 明
close()	立即释放此 Statement 对象的数据库和 JDBC 资源
excute(String sql)	执行给定的 SQL 语句，该语句可能返回多个结果
excuteBatch()	将一批命令提交给数据库来执行，如果全部命令执行成功，则返回更新计数组成的数组
excuteQuery(String sql)	执行给定的 SQL 语句，该语句返回单个 ResultSet 对象
excuteUpdate(String sql, int autoGeneratedKeys)	执行给定的 SQL 语句，并用给定标志通知驱动程序由此 Statement 生成的自动生成键是否可用于检索
excuteUpdate(String sql)	执行给定 SQL 语句，该语句可能为 INSERT、UPDATE 或 DELETE 语句，或者不返回任何内容的 SQL 语句（如 SQL DDL 语句）
getFetchSize()	检索结果集合的行数，该数是根据此 Statement 对象生成的 ResultSet 对象的默认获取大小
getResultSet()	以 ResultSet 对象的形式检索当前结果

【例 10.3】 批量插入记录。

实例位置：光盘 \MR\ 源码 \ 第 10 章 \10.3

本实例在程序的主页面 index.jsp 中，向数据库中批量插入数据，首先定义一个 String 型数组，用来存放插入记录的名称，代码如下：

```
String[] names = { "A", "B", "C" };                    // 定义要插入的数据
```

然后创建数据库连接对象，代码如下：

```
Class.forName("com.mysql.jdbc.Driver");
Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/db_
database10","root","111");
```

最后再通过一次批量操作在数据库中添加 3 条记录，通过循环遍历上面的 String 型数组动态生成 INSERT 语句，代码如下：

```
try {
    Statement stmt = conn.createStatement();      // 创建 Statement 对象
    stmt.clearBatch();                            // 清除位于 Batch 中的所有 SQL 语句
    for (int i = 0; i < names.length; i++) {
    // 通过循环向 Batch 中添加欲执行的 SQL 语句
        stmt.addBatch("insert into tb_batch(name) values('" + names[i] + "')");
    }
    stmt.executeBatch();                          // 批量执行 Batch 中的 SQL 语句
    stmt.close();
    conn.close();
    out.println(" 批量添加成功！ ");
} catch (SQLException e) {
   out.println(" 批量添加失败！ ");
    e.printStackTrace();
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

程序运行后，将在数据表中插入如图 10.13 所示的数据。



图 10.13　在数据表中插入的数据

10.3.4　预编译执行 SQL 语句的 PreparedStatement 接口

Statement 接口封装了 JDBC 执行 SQL 语句的方法，可以完成 Java 程序执行 SQL 语句的操作。在实际开发过程中往往需要将程序中的变量作为 SQL 语句的查询条件等，使用 Statement 接口操作这些 SQL 语句会过于烦琐，而且存在安全方面的缺陷。针对这一问题，JDBC API 提供了扩展的 PreparedStatement 接口。

PreparedStatement 接口继承于 Statement 接口，拥有其中的方法。PreparedStatement 对象保存并执行的 SQL 语句是预编译的，可以使用此对象高效地多次执行 SQL 语句。而且该接口扩展了带有参数 SQL 语句的执行操作，应用该接口中的 SQL 语句可以使用占位符“?”来代替其参数，然后通过 setXXX() 方法为 SQL 语句的参数赋值。该接口的常用方法如表 10.4 所示。

表 10.4　PreparedStatement 接口的常用方法

方　法	说　明
execute()	在此 PreparedStatement 对象中可以执行任何 SQL 语句
executeQuery()	在此 PreparedStatement 对象中执行 SQL 查询并返回该查询生成的 ResultSet 对象
executeUpdate()	在此 PreparedStatement 对象中执行 SQL 语句，该语句必须是一个 SQL INSERT、UPDATE 或 DELETE 语句，也可以是不返回结果的 SQL 语句，如 DDL 语句
setBoolean(int parameterIndex, boolean x)	将指定参数设置为给定 Java boolean 值
setByte(int parameterIndex, byte x)	将指定参数设置为给定 Java byte 值
setDate(int parameterIndex, Date x)	将指定参数设置为给定 java.sql.Date 值
setInt(int parameterIndex, int x)	将指定参数设置为给定 Java int 值
setShort(int parameterIndex, short x)	将指定参数设置为给定 Java short 值
setString(int parameterIndex, Strng x)	将指定参数设置为给定 Java String 值

注意　在通过 setXXX() 方法为 SQL 语句中的参数赋值时，必须通过与输入参数的已定义 SQL 类型兼容的方法，也可以通过 setObject() 方法设置多种类型的输入参数。

PreparedStatement 的使用方法如下。

```
PreparedStatement ps = connection.prepareStatement("select * from table_name
where id>? and (name=? or name=?)");
ps.setInt(1, 6);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
ps.setString(2, "B");
ps.setObject(3, "D");
ResultSet rs = ps.executeQuery();
```

10.3.5 结果集 ResultSet 接口

ResultSet 接口用于保存 JDBC 执行查询时返回的结果集，该结果集与数据库表字段相对应。即由行和列组成，并且在 ResultSet 结果集的行上提供指针。最初指针指向结果集的第 1 行之前，调用 next() 方法可将指针移动到下一行。如果下一行没有数据，则返回 false。在应用程序中经常使用 next() 方法作为 While 循环的条件来迭代 ResultSet 结果集。图 10.14 所示为 ResultSet 接口的原理结构。

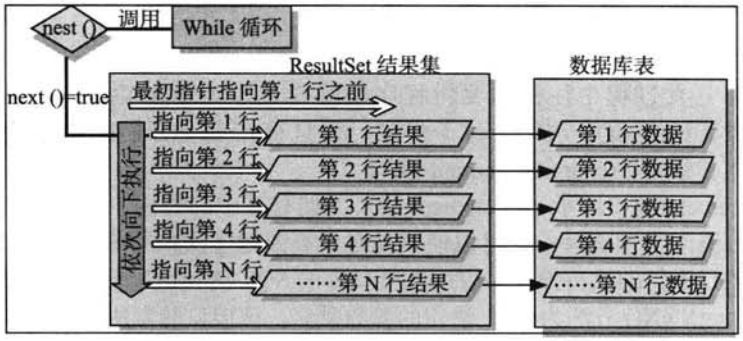


图 10.14 ResultSet 接口的原理结构

ResultSet 接口的常用方法如表 10.5 所示。

表 10.5 ResultSet 接口的常用方法

方 法	说 明
absolute(int row)	将指针移动到此 ResultSet 对象的给定行编号
afterLast()	将指针移动到此 ResultSet 对象的末尾，位于最后一行之后
beforeFirst()	将指针移动到此 ResultSet 对象的开头，位于第 1 行之前
close()	立即释放此 ResultSet 对象的数据库和 JDBC 资源
first()	将指针移动到此 ResultSet 对象的第 1 行
getBoolean(int columnIndex)	用 boolean 的形式检索 ResultSet 对象的当前行中指定列的值
getByte(int columnIndex)	用 byte 的形式检索 ResultSet 对象的当前行中指定列的值
getBytes(int columnIndex)	用 byte 数组的形式检索 ResultSet 对象的当前行中指定列的值
getDate(int columnIndex)	用 java.sql.Date 对象的形式检索 ResultSet 对象的当前行中指定列的值
getFloat(int columnIndex)	用 float 的形式检索 ResultSet 对象的当前行中指定列的值
getInt(int columnIndex)	用 int 的形式检索 ResultSet 对象的当前行中指定列的值
getString(int columnIndex)	用 String 的形式检索 ResultSet 对象的当前行中指定列的值
next()	将指针从当前位置下移一行，常用于 While 循环的迭代条件

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明 ResultSet 对象的 get<Type>() 方法中的 int 类型参数索引从 1 开始，如表的第 1 列字段名为“id”，字段类型为 int，那么应该使用 getInt(1) 索引该列的值。也可以使用字段名索引，即 getInt(“id”)。

例如，下面的代码可以通过 ResultSet 接口获得数据表中每一列的数据并输出到网页。

```
ResultSetMetaData rsmd=rs.getMetaData();
for(int i=1;i<=rsmd.getColumnCount();i++){
     out.println("<td>"+rsmd.getColumnName(i)+"</td>");
}
```

通过 ResultSet 接口也可以操纵数据表，例如下面的代码。

```
//创建 Statement 实例，通过该实例创建的 ResultSet 实例将具有给定类型和并发性
stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_
UPDATABLE);
//下面的代码修改第 3 条记录
rs = stmt.executeQuery("select * from tb_iud");
rs.absolute(3);                                    // 将指针移至第 3 条记录
rs.updateString(2, "update");                      // 将第 2 个字段修改为 update
rs.updateRow();                                    // 将修改操作同步到数据库
//下面的代码插入第 4 条记录
rs = stmt.executeQuery("select * from tb_iud");
rs.moveToInsertRow();                              // 将指针移至插入行
rs.updateInt(1, 4);                                // 设置要插入行的第 1 个字段
rs.updateString(2, "insert");                      // 将第 2 个字段修改为 insert
rs.insertRow();                                    // 将插入操作同步到数据库
//下面的代码删除第 1 条记录
rs = stmt.executeQuery("select * from tb_iud");
rs.next();                                         // 将指针移至下一条记录，即第 1 条
rs.deleteRow();                                    // 删除该记录
rs.close();                                        // 将删除操作同步到数据库
```

注意 在执行 deleteRow() 方法时并不将操作同步到数据库，而是在关闭 ResultSet 结果集时同步到数据库。

10.3.6　及时关闭连接

每次操纵数据库后都要及时关闭数据库连接，释放连接占用的数据库和 JDBC 资源，以免影响系统的运行速度。

ResultSet、Statement 和 Connection 接口均提供了关闭各自实例的 close() 方法以释放各自占用的数据库和 JDBC 资源。建议在每次操纵数据库后依次关闭 ResultSet、Statement 和 Connection 实例，虽然直接关闭 Statement 实例也会关闭相关的 ResultSet 实例，并且 JVM 的垃圾回收机制会定时清理缓存并关闭长时间不用的数据库连接。但是如果不及时关闭，当数据库连接达到一定数量，将严重影响数据库和计算机的运行速度，甚至使数据库瘫痪。关闭各自实例的方法如下。

```
resultSet.close();                    // 释放 ResultSet 占用的资源
statement.close();                    // 释放 Statement 占用的资源
connection.close();                   // 释放 Connection 占用的资源
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

10.4 通过JDBC操作数据库

本节讲解如何通过 JDBC API 来操作数据库，包括数据库的增、删、改、查及批处理等。

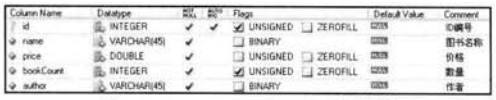
10.4.1 添加数据

要通过 JDBC 在数据库中添加数据，可以使用 INSERT 语句，SQL 语句中的参数可以用占位符“?”代替，然后通过 PreparedStatement 为其赋值及并执行 SQL 语句。

【例 10.4】 创建 Web 项目，通过 JDBC 实现添加图书信息的功能。

实例位置：光盘\MR\源码\第 10 章\10.4

Step 1) 在 MySQL 数据库中创建图书信息表 tb_books，其结构如图 10.15 所示。



Column Name	Datatype	PK	NN	Flags	Default Value	Comment
id	INTEGER	✓	✓	UNSIGNED ZEROFILL		编号
name	VARCHAR(45)	✓		BINARY		图书名称
price	DOUBLE	✓		UNSIGNED ZEROFILL		价格
bookCount	INTEGER	✓		UNSIGNED ZEROFILL		数量
author	VARCHAR(45)	✓		BINARY		作者

图 10.15 tb_books 表的结构

Step 2) 创建名为 Book 的 JavaBean，用于封装图书对象信息，其关键代码如下：

```
public class Book {
    private int id;                          // 编号
    private String name;                     // 图书名称
    private double price;                    // 价格
    private int bookCount;                   // 数量
    private String author;                   // 作者
    public int getId() {
            return id;
    }
    public void setId(int id) {
            this.id = id;
    }
    // 省略部分 Setter 与 Getter 方法
}
```

Step 3) 创建 index.jsp 页面作为程序中的主页，用于放置添加图书信息所需要的表单。此表单提交到 AddBook.jsp 页面处理，其关键代码如下：

```
<form action="AddBook.jsp" method="post" onsubmit="return check(this);">
            <h2>添加图书信息</h2>
            图书名称：<input type="text" name="name" />
            价    格：<input type="text" name="price" />
            数    量：<input type="text" name="bookCount" />
            作    者：<input type="text" name="author" />
            <input type="submit" value="添  加">
</form>
```

Step 4) 创建 AddBook.jsp 页面，用于处理添加图书信息的请求。此页面通过 JDBC 所提交的图书信息数据写入数据库中，其关键代码如下：

```
<%request.setCharacterEncoding("UTF-8"); %>
<jsp:useBean id="book" class="com.lyq.bean.Book"></jsp:useBean>
<jsp:setProperty property="*" name="book"/>
<%
    try {
            Class.forName("com.mysql.jdbc.Driver");  // 加载数据库驱动，注册到驱动管理器
            String url = "jdbc:mysql://localhost:3306/db_database10";
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                                                    // 数据库连接字符串
            String username = "root";               // 数据库用户名
            String password = "111";                // 数据库密码
            Connection conn = DriverManager.getConnection(url,username,password);
                                                    // 创建 Connection 连接
            // 添加图书信息的 SQL 语句
            String sql = "insert into tb_books(name,price,bookCount,author)
            values(?,?,?,?)";
            PreparedStatement ps = conn.prepareStatement(sql);
                                                    // 获取 PreparedStatement
            ps.setString(1, book.getName());        // 对 SQL 语句中的第 1 个参数赋值
            ps.setDouble(2, book.getPrice());       // 对 SQL 语句中的第 2 个参数赋值
            ps.setInt(3,book.getBookCount());       // 对 SQL 语句中的第 3 个参数赋值
            ps.setString(4, book.getAuthor());      // 对 SQL 语句中的第 4 个参数赋值
            int row = ps.executeUpdate();           // 执行更新操作，返回所影响的行数
            if(row > 0){                                            // 判断是否更新成功
                    out.print("成功添加了 " + row + "条数据！ ");    // 更新成输出信息
            }
            ps.close();                             // 关闭 PreparedStatement，释放资源
            conn.close();                           // 关闭 Connection，释放资源
    } catch (Exception e) {
            out.print("图书信息添加失败！ ");
            e.printStackTrace();
    }
%>
<br>
<a href="index.jsp">返回 </a>
```

在 AddBook.jsp 页面中首先通过 <jsp:useBean> 实例化 JavaBean 对象 Book，并通过 <jsp:setProperty> 为 Book 对象中的属性赋值，在构建图书对象后通过 JDBC 将图书信息写入数据库中。

技巧 <jsp:setProperty> 标签的 property 属性的值可以设置为“*”，作用是将与表单中同名的属性值赋给 JavaBean 对象中的同名属性。使用此种方式不必为 JavaBean 中的属性一一赋值，从而减少代码量。

在数据库插入图书信息主要通过 PreparedStatement 对象操作，使用该对象，其 SQL 语句中的参数可以使用占位符“?”代替。然后通过 PreparedStatement 对象为 SQL 语句中的参数逐一赋值，将图书信息传递到 SQL 语句中。

注意 使用 PreparedStatement 对象为 SQL 语句的占位符参数赋值，第 1 个参数的下标值从 1 开始，与数组的下标有所区别。

通过 PreparedStatement 对象为 SQL 语句中的参数赋值后，还不能将图书信息写入到数据库中，需要调用其 executeUpdate() 方法执行更新操作。此方法执行后返回的 int 型数据是所影响的行数，实例中获取该值并输出到页面中。

技巧 在操作数据库后应该立即调用 ResultSet、PreparedStatement 和 Connection 对象的 close() 方法及时释放占用的数据库资源。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Step 5) 编写 AddBook.jsp 页面后，部署并运行程序。打开添加图书信息页面，如图 10.16 所示。

正确填写图书信息后，单击“添加”按钮，图书信息数据将被写入到数据库中。页面输出提示信息，如图 10.17 所示。

成功添加图书信息后打开数据表 tb_books，可以查看到添加的图书信息，如图 10.18 所示。



图 10.16 “添加图书信息”页面

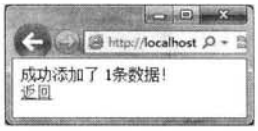


图 10.17 提示信息



id	name	price	bookCount	author
4	Java Web开发实战宝典	89	10	明日科技
5	Java Web开发实战1200例	96	10	明日科技

图 10.18 添加的图书信息

说明 由于 id 值设置了自动编号，所以数据库会自动生成添加的图书信息中的 id，数据表 tb_books 中的最后一条数据为添加的数据。

10.4.2 查询数据

使用 JDBC 查询数据与添加数据的流程基本相同，但需要通过 ResultSet 对象来加载查询结果集。

ResultSet 对象是 JDBC API 中封装的结果集对象，从数据表中查询的数据均放置在这个集合中，其结构如图 10.19 所示。



图 10.19 ResultSet 的结构

在 ResultSet 集合中通过移动“光标”来获取查询的数据，ResultSet 对象中的“光标”可以上下移动。如获取 ResultSet 集合中的一条数据，只需要把“光标”定位到该数据行即可。

注意 从图 10.19 可以看出 ResultSet 集合中的查询数据位于集合的中间位置，在第 1 条数据之前与最后一条数据后有一个位置。默认情况下，ResultSet 的光标位置在第 1 行数据之前，所以在第 1 次获取数据时需要移动光标位置。

【例 10.5】 创建 Web 项目，通过 JDBC 查询图书信息表中的图书信息并显示在 JSP 页面中。

实例位置：光盘 \MR\ 源码 \ 第 10 章 \10.5

Step 1) 创建名为 Book 的 JavaBean，用于封装图书信息。由于该 JavaBean 的代码与例 10.4 的完全相同，所以这里不再赘述。

Step 2) 创建名为 FindServlet 的 Servlet 对象，用于查询所有图书信息。在其中编写 doGet() 方法建立数据库连接，并将查询的数据集合放置到 HttpServletRequest 对象中，将请求转发到 JSP 页面，其关键代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    try {
        Class.forName("com.mysql.jdbc.Driver");  // 加载数据库驱动，注册到驱动管理器
        String url = "jdbc:mysql://localhost:3306/db_database10";
                                                 // 数据库连接字符串
        String username = "root";                // 数据库用户名
        String password = "111";                 // 数据库密码
        Connection conn = DriverManager.getConnection(url,username,password);
                                                 // 创建 Connection 连接
        Statement stmt = conn.createStatement();  // 获取 Statement
        String sql = "select * from tb_book";     // 添加图书信息的 SQL 语句
        ResultSet rs = stmt.executeQuery(sql);    // 执行查询
        List<Book> list = new ArrayList<Book>();  // 实例化 List 对象
        while(rs.next()){                         // 光标向后移动，并判断是否有效
                Book book = new Book();           // 实例化 Book 对象
                book.setId(rs.getInt("id"));      // 为 id 属性赋值
                book.setName(rs.getString("name"));  // 为 name 属性赋值
                book.setPrice(rs.getDouble("price"));  // 为 price 属性赋值
                book.setBookCount(rs.getInt("bookCount"));
                                                  // 为 bookCount 属性赋值
                book.setAuthor(rs.getString("author"));  // 为 author 属性赋值
                list.add(book);                   // 将图书对象添加到集合中
        }
        request.setAttribute("list", list);      // 将图书集合放置到 request 之中
        rs.close();                               // 关闭 ResultSet
        stmt.close();                             // 关闭 Statement
        conn.close();                             // 关闭 Connection
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    request.getRequestDispatcher("book_list.jsp").forward(request, response);
                                                  // 请求转发到 book_list.jsp
}
```

说明　由于查询图书信息的 SQL 语句中并不需要参数信息，所以实例中使用 Statement 对象执行查询。

在 doGet() 方法中首先获取数据库的连接 Connection，然后通过 Statement 对象执行查询图书信息的 SELECT 语句，并获取 ResultSet 结果集。最后遍历 ResultSet 的中数据来封装图书对象 Book，将其添加到 List 集合中，转发到显示页面显示。

说明　ResultSet 集合中的第 1 行数据之前与最后一行数据之后都存在一个位置，默认情况下光标位于第 1 行数据之前。使用 Java 之中的 for 循环和 do...while 循环等不能完整地遍历数据，所以实例中使用 while 条件循环遍历 ResultSet 对象。在第 1 次循环时执行条件 rs.next()，将光标移动到第 1 条数据的位置。

获取 ResultSet 对象后，即可通过移动光标定位到查询结果中的指定行，然后通过 ResultSet 对象提供的一系列 Getter 方法来获取当前行的数据。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

注意 使用 ResultSet 对象提供的 Getter 方法获取的数据，其类型要与数据表中字段类型相对应，否则将抛出 java.sql.SQLExcption 异常。

Step 3 创建 book_list.jsp 页面，用于显示所有图书信息，其关键代码如下：

```
<table align="center" width="450" border="1">
   <tr>
          <td align="center" colspan="5">
                 <h2> 所有图书信息 </h2>
          </td>
   </tr>
   <tr align="center">
          <td><b>ID</b></td>
          <td><b> 图书名称 </b></td>
          <td><b> 价格 </b></td>
          <td><b> 数量 </b></td>
          <td><b> 作者 </b></td>
   </tr>
          <%
                 // 获取图书信息集合
                        List<Book> list = (List<Book>)request.
                        getAttribute("list");
                        // 判断集合是否有效
                        if(list == null || list.size() < 1){
                               out.print(" 没有数据！ ");
                        }else{
                               // 遍历图书集合中的数据
                               for(Book book : list){
          %>
                 <tr align="center">
                        <td><%=book.getId()%></td>
                        <td><%=book.getName()%></td>
                        <td><%=book.getPrice()%></td>
                        <td><%=book.getBookCount()%></td>
                        <td><%=book.getAuthor()%></td>
                 </tr>
          <%
                        }
                 }
          %>
</table>
```

由于 FindServlet 将查询的所有图书信息集合放到了 request 之中，所以在 book_list.jsp 页面中可以通过 request 的 getAttribute() 方法获取这一集合对象。实例中在获取所有图书信息集合后通过 for 循环遍历所有图书信息集合，并将其输出到页面之中。

技巧 在 book_list.jsp 页面中使用 for...in 循环遍历所有图书信息，此种方式可以简化程序。

Step 4 创建 index.jsp 页面作为程序中的主页，在其中编写一个导航链接用于请求查看所有图书信息，其关键代码如下：

```
<body>
   <a href="FindServlet"> 查看所有图书 </a>
</body>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

部署并运行程序后打开 index.jsp 页面，单击“查看所有图书”超链接可以查看从数据库中查询的所有图书信息，运行结果如图 10.20 所示。



图 10.20　运行结果

10.4.3　修改数据

使用 JDBC 修改数据库中的数据可以通过 UPDATE 语句实现，如把图书 id 为 1 的图书数量修改为 100，其 SQL 语句如下：

```
update tb_book set bookcount=100 where id=1
```

在实际开发中通常都是由程序传递 SQL 语句中的参数，所以修改数据也需要使用 PreparedStatement 对象执行操作。

【例 10.6】 在查询所有图书信息的页面中添加修改图书数量的表单，通过 Servlet 修改数据库中的图书数量。

实例位置：光盘 \MR\ 源码 \ 第 10 章 \10.6

Step 1) 在 book_list.jsp 页面中增加修改图书数量的表单，将此表单的提交地址设置为 UpdateServlet，其关键代码如下：

```
<table align="center" width="500" border="1">
    <tr>
        <td align="center" colspan="6">
            <h2> 所有图书信息 </h2>
        </td>
    </tr>
    <tr align="center">
        <td><b>ID</b></td>
        <td><b> 图书名称 </b></td>
        <td><b> 价格 </b></td>
        <td><b> 数量 </b></td>
        <td><b> 作者 </b></td>
        <td><b> 修改数量 </b></td>
    </tr>
        <%
            // 获取图书信息集合
            List<Book> list = (List<Book>)request.getAttribute("list");
            // 判断集合是否有效
            if(list == null || list.size() < 1){
                out.print(" 没有数据！ ");
            }else{
                // 遍历图书集合中的数据
                for(Book book : list){
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        %>
                <tr align="center">
                        <td><%=book.getId()%></td>
                        <td><%=book.getName()%></td>
                        <td><%=book.getPrice()%></td>
                        <td><%=book.getBookCount()%></td>
                        <td><%=book.getAuthor()%></td>
                        <td>
                                <form action="UpdateServlet" method="post"
                                 onsubmit="return  check(this);">
                                        <input type="hidden" name="id"
                                         value="<%=book.getId()%>">
                                        <input type="text" name="bookCount"
                                         size="3">
                                        <input type="submit" value="修  改">
                                </form>
                        </td>
                </tr>
        <%
                        }
                }
        %>
</table>
```

在修改图书信息的表单中主要包含两个属性信息，分别为图书 id 与图书数量 bookCount，这是因为修改图书数量时需要明确指定图书的 id 作为修改条件，否则将会修改所有图书信息记录。

技巧 由于图书 id 属性并不需要显示在表单之中，而在图书信息的修改过程中需要获取这个值，所以将 id 对应文本框 <input> 中的 type 属性设置为 hidden。使其在表单之中构成一个隐藏域，从而实现实际的业务需求。

Step 2 创建修改图书信息请求的 Servlet 对象，其名为 UpdateServlet。由于表单提交的请求类型为 post，所以在 UpdateServlet 中编写 doPost() 方法处理修改图书信息的请求，其关键代码如下：

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    int id = Integer.valueOf(request.getParameter("id"));
    int bookCount = Integer.valueOf(request.getParameter("bookCount"));
    try {
            Class.forName("com.mysql.jdbc.Driver");
                                                    // 加载数据库驱动，注册到驱动管理器
            String url = "jdbc:mysql://localhost:3306/db_database10";
                                                    // 数据库连接字符串
            String username = "root";               // 数据库用户名
            String password = "111";                // 数据库密码
            Connection conn = DriverManager.getConnection(url,username,password);
                                                    // 创建 Connection 连接
            String sql = "update tb_book set bookcount=? where id=?";
                                                    // 更新 SQL 语句
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            PreparedStatement ps = conn.prepareStatement(sql);
                                                    // 获取 PreparedStatement
            ps.setInt(1, bookCount);                // 对 SQL 语句中的第 1 个参数赋值
            ps.setInt(2, id);               // 对 SQL 语句中的第 2 个参数赋值
            ps.executeUpdate();             // 执行更新操作
            ps.close();                     // 关闭 PreparedStatement
            conn.close();                   // 关闭 Connection
        } catch (Exception e) {
            e.printStackTrace();
        }

        response.sendRedirect("FindServlet");   // 重定向到 FindServlet
    }
```

说明　HttpServletRequest 接受的参数值为 String 类型，而图书 id 与图书数量为 int 类型，所以需要执行转换操作，实例中通过 Integer 类的 valueOf() 方法实现。

在 UpdateServlet 的 doPost() 方法中首先通过 HttpServletRequest 获取图书的 id 与修改的图书数量，然后建立数据库连接 Connection。通过 PreparedStatement 对 SQL 语句进行预处理并为 SQL 语句参数赋值，最后执行更新操作。

注意　在执行更新操作之后一定要关闭数据库连接，以及时释放所占用的数据库资源。

在执行图书数量的更新操作后，实例中通过 HttpServletRequest 对象将请求重定向到 FindServlet，查看更新后的结果。

实例运行后打开首页，单击“查看所有图书”超链接，打开图书信息列表页面。在该页面中，正确填写图书数量后，如图 10.21 所示，单击“修改”按钮即可将图书数量更新到数据之中。



图 10.21　修改图书数量

10.4.4　删除数据

删除数据使用的 SQL 语句为 DELETE 语句，如删除图书 id 为 1 的图书信息的 SQL 语句如下：

```
delete from tb_book where id=1
```

在实际开发中，通常都是由程序传递 SQL 语句中的参数，所以修改数据也需要使用 PreparedStatement 对象操作。

【例 10.7】 在查询所有图书信息的页面中添加删除图书信息的超链接，通过 Servlet 删除数据。

实例位置：光盘 \MR\ 源码 \ 第 10 章 \10.7

Step 1) 在 book_list.jsp 页面中增加删除图书信息的超链接，将其地址指向 DeleteServlet，关键代码如下：

```
<table align="center" width="500" border="1">
    <tr>
        <td align="center" colspan="6">
            <h2> 所有图书信息 </h2>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            </td>
        </tr>
        <tr align="center">
            <td><b>ID</b></td>
            <td><b> 图书名称 </b></td>
            <td><b> 价格 </b></td>
            <td><b> 数量 </b></td>
            <td><b> 作者 </b></td>
            <td><b> 删除 </b></td>
        </tr>
            <%
                List<Book> list = (List<Book>)request.getAttribute("list");
                                                        // 获取图书信息集合
                if(list == null || list.size() < 1){    // 判断集合是否有效
                    out.print(" 没有数据！ ");
                }else{
                    for(Book book : list){              // 遍历图书集合中的数据
            %>
                <tr align="center">
                    <td><%=book.getId()%></td>
                    <td><%=book.getName()%></td>
                    <td><%=book.getPrice()%></td>
                    <td><%=book.getBookCount()%></td>
                    <td><%=book.getAuthor()%></td>
                    <td>
                        <a href="DeleteServlet?id=<%=book.getId()%>"> 删除 </a>
                    </td>
                </tr>
            <%
                    }
                }
            %>
    </table>
```

在删除数据操作中需要传递要删除的图书对象，因此在删除图书信息的超链接中加入图书 id 值。

技巧 在 Java Web 开发中，JSP 页面的超链接可以带参数，通过在超链接后添加“?”来实现。

Step 2 编写处理删除图书信息的 Servlet，其名为 DeleteServlet。在 doGet() 方法中编写删除图书信息的方法，其关键代码如下：

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    int id = Integer.valueOf(request.getParameter("id")); // 获取图书 id
    try {
        Class.forName("com.mysql.jdbc.Driver"); // 加载数据库驱动，注册到驱动管理器
        String url = "jdbc:mysql://localhost:3306/db_database10";
                                                        // 数据库连接字符串
        String username = "root";                       // 数据库用户名
        String password = "111";                        // 数据库密码
        Connection conn = DriverManager.getConnection(url,username,password);
                                                        // 创建 Connection 连接
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            String sql = "delete from tb_book where id=?"; // 删除图书信息的SQL语句
            PreparedStatement ps = conn.prepareStatement(sql);
                                                    // 获取PreparedStatement
            ps.setInt(1, id);                       // 对SQL语句中的第1个占位符赋值
            ps.executeUpdate();                     // 执行更新操作
            ps.close();                             // 关闭PreparedStatement
            conn.close();                           // 关闭Connection
        } catch (Exception e) {
            e.printStackTrace();
        }
        response.sendRedirect("FindServlet");       // 重定向到FindServlet
    }
```

在 DeleteServlet 类的 doGet() 方法中首先获取要删除的图书 id 值，然后创建数据库连接 Connection。通过 PreparedStatement 对 SQL 语句进行预处理并为 SQL 语句参数赋值，最后执行删除操作。

技巧 在 DeleteServlet 完成删除数据操作后，通过 HttpServletResponse 对象将请求重定向到 FindServlet 再次执行查询所有图书信息操作，以查看删除后的结果。

实例运行后打开首页，单击“查看所有图书”超链接打开图书信息列表页面。在其中可以看到每一条图书信息的超链接，如图 10.22 所示。单击每一条图书信息对应的“删除”超链接后此条图书信息将从数据库中删除。



图 10.22　每一条图书信息的超链接

10.4.5　批处理

在 JDBC 开发中操作数据库需要与数据库建立连接，然后将要执行的 SQL 语句传送到数据库服务器，最后关闭数据库连接。如果按照此流程执行多条 SQL 语句，那么就需要建立多个数据库连接。针对这一问题，JDBC 的批处理提供了很好的解决方案。

JDBC 批处理的原理是批量将 SQL 语句一次性发送到数据库中执行，从而解决多次与数据库连接所产生的速度瓶颈。

【例 10.8】 创建学生信息表，通过 JDBC 的批处理操作一次性将多个学生信息写入到数据库中。

实例位置：光盘 \MR\ 源码 \ 第 10 章 \10.8

Step 1) 创建学生信息表 tb_student_batch 表，其结构如图 10.23 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INTEGER	✔		☑ UNSIGNED ☐ ZEROFILL	NULL	学号
name	VARCHAR(45)	✔		☐ BINARY	NULL	姓名
sex	TINYINT(1)	✔		☐ UNSIGNED ☐ ZEROFILL	NULL	性别
age	INTEGER	✔		☑ UNSIGNED ☐ ZEROFILL	NULL	年龄

图 10.23　学生信息表 tb_student_batch 的结构

Step 2 创建名为 Batch 的类，用于实现添加学生信息的批量操作。在 Batch 类中编写 getConnection() 方法，用于获取数据库连接 Connection 对象，其关键代码如下：

```
/**
 * 获取数据库连接
 * @return Connection 对象
 */
public Connection getConnection(){
 Connection conn = null;                                    // 数据库连接
 try {
      Class.forName("com.mysql.jdbc.Driver");    // 加载数据库驱动，注册到驱动管理器
      String url = "jdbc:mysql://localhost:3306/db_database10";
                                                            // 数据库连接字符串
      String username = "root";                             // 数据库用户名
      String password = "111";                              // 数据库密码
      conn = DriverManager.getConnection(url,username,password);
                                                            // 创建 Connection 连接
 } catch (ClassNotFoundException e) {
      e.printStackTrace();
 } catch (SQLException e) {
      e.printStackTrace();
 }
 return conn;                                               // 返回数据库连接
}
```

Step 3 编写 saveBatch() 方法，实现批量添加学生信息功能，实例中主要通过 PreparedStatement 对象实现，其关键代码如下：

```
/**
 * 批量添加数据
 * @return 影响的行数
 */
public int saveBatch(){
 int row = 0 ;                                              // 行数
 Connection conn = getConnection();                         // 获取数据库连接
 try {
      String sql = "insert into tb_student_batch(id,name,sex,age) values (?,?,?,?)";
                                                            // 插入数据的 SQL 语句
      PreparedStatement ps = conn.prepareStatement(sql);
                                                            // 创建 PreparedStatement
      Random random = new Random();                         // 实例化 Random
      for (int i = 0; i < 10; i++) {                        // 循环添加数据
            ps.setInt(1, i+1);                              // 为 SQL 语句中的第 1 个参数赋值
            ps.setString(2, "学生" + i);                    // 为 SQL 语句中的第 2 个参数赋值
            ps.setBoolean(3, i % 2 == 0 ? true:false);
                                                            // 为 SQL 语句中的第 3 个参数赋值
            ps.setInt(4, random.nextInt(5) + 10);
                                                            // 为 SQL 语句中的第 4 个参数赋值
            ps.addBatch();                                  // 添加批处理命令
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        }
        int[] rows = ps.executeBatch();                    // 执行批处理操作并返回计数组成的数组
        row = rows.length;                                 // 对行数赋值
        ps.close();                                        // 关闭 PreparedStatement
        conn.close();                                      // 关闭 Connection
    } catch (Exception e) {
        e.printStackTrace();
    }
    return row;                                            // 返回添加的行数
    }
```

实例中在创建 PreparedStatement 对象后，通过 for 循环在 PreparedStatement 中批量添加 SQL 命令，其中学生信息数据通过程序模拟生成。

技巧　实例中为学生性别字段赋值为布尔类型，如果变量 i 整除 2，则为 true，否则为 false。此操作通过“?:”的语句实现，它相当于 if...else 语句，使用此种代码可以简化程序。

执行批处理操作后，实例中获取返回计数组成的数组，将数组的长度赋值给 row 变量来计算数据库操作所影响的行数。

注意　PreparedStatement 对象的批处理操作调用的是 executeBatch() 方法，而不是 execute() 或 executeUpdate() 方法。

Step 4　创建程序中的首页面 index.jsp，在其中通过 <jsp:useBean> 实例化 Batch 对象并执行批量添加数据操作，其关键代码如下：

```
<jsp:useBean id="batch" class="com.lyq.Batch"></jsp:useBean><%
    // 执行批量插入操作
    int row = batch.saveBatch();
    out.print("批量插入了【" + row + "】条数据！ ");
%>
```

实例运行后在数据库中批量添加了 10 条学生数据，运行结果如图 10.24 所示。

运行成功后，可以打开数据表 tb_student_batch 查看，如图 10.25 所示。

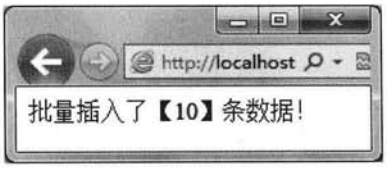


图 10.24　运行结果

	id	name	sex	age
▶	1	学生0	1	10
	2	学生1	0	14
	3	学生2	1	11
	4	学生3	0	10
	5	学生4	1	14
	6	学生5	0	10
	7	学生6	1	14
	8	学生7	0	11
	9	学生8	1	13
	10	学生9	0	10

图 10.25　tb_student_batch 表中的数据

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

10.4.6 调用存储过程

在 JDBC API 中提供了通过 CallableStatement 对象调用存储过程的方法，该对象位于 java.sql 包中。它继承于 Statement 对象，主要用于执行数据库中定义的存储过程，其调用方法如下：

```
{call <procedure-name>[(<arg1>,<arg2>, ...)]}
```

其中 arg1 和 arg2 为存储过程中的参数，如果存储过程中需要传递参数，可以为其赋值。

说明 存储过程是一个 SQL 语句和可选控制流语句的预编译集合，编译完成后存放在数据库中，这样就省去了执行 SQL 语句时编译 SQL 语句所花费的时间。在执行存储过程时只需要将参数传递到数据库中，而不需要将整条 SQL 语句提交给数据库，这样减少了网络传输的流量，也提高了程序的运行速度。

【例 10.9】 创建查询所有图书信息的存储过程，通过 JDBC API 获取所有图书信息，并输出到 JSP 页面中。

实例位置：光盘 \MR\ 源码 \ 第 10 章 \10.9

Step 1) 在数据库 db_database10 中创建名为 findAllBook 的存储过程，用于查询所有图书信息，其关键代码如下：

```
DELIMITER $$

CREATE PROCEDURE 'db_database10'.'findAllBook' ()
BEGIN
    SELECT * FROM tb_book ORDER BY id DESC;
END
```

说明 各种数据库创建存储过程并非一致，本实例使用 MySQL 数据库。如使用其他数据库创建存储过程请求，则参阅相应的帮助文档。

Step 2) 创建名为 Book 的 JavaBean，用于封装图书信息，由于该 JavaBean 的代码和例 10.4 完全相同，这里就不再赘述。

Step 3) 创建名为 FindBook 的类，用于执行查询图书信息的存储过程。首先在其中编写 getConnection() 方法，获取数据库连接对象 Connection，其关键代码如下：

```
/**
 * 获取数据库连接
 * @return Connection 对象
 */
public Connection getConnection(){
 Connection conn = null;                                    // 数据库连接
 try {
       Class.forName("com.mysql.jdbc.Driver");  // 加载数据库驱动，注册到驱动管理器
       String url = "jdbc:mysql://localhost:3306/db_database10";
                                                            // 数据库连接字符串
       String username = "root";                            // 数据库用户名
       String password = "111";                             // 数据库密码
       conn = DriverManager.getConnection(url,username,password);
                                                            // 创建 Connection 连接
  } catch (ClassNotFoundException e) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return conn;                                    // 返回数据库连接
    }
```

然后编写 findAll() 方法，调用数据库中定义的存储过程 findAllBook 查询所有图书信息并将查询结果放置到 List 集合中，其关键代码如下：

```
/**
 * 通过存储过程查询数据
 * @return  List<Book>
 */
public List<Book> findAll(){
    List<Book> list = new ArrayList<Book>();                    // 实例化 List 对象
    Connection conn = getConnection();                          // 创建数据库连接
    try {
        CallableStatement cs = conn.prepareCall("{call findAllBook()}");
                                                                // 调用存储过程
        ResultSet rs = cs.executeQuery();                       // 执行查询操作，并获取结果集
        while(rs.next()){                                   // 光标向后移动，并判断是否有效
            Book book = new Book();                             // 实例化 Book 对象
            book.setId(rs.getInt("id"));                        // 为 id 属性赋值
            book.setName(rs.getString("name"));                 // 为 name 属性赋值
            book.setPrice(rs.getDouble("price"));               // 为 price 属性赋值
            book.setBookCount(rs.getInt("bookCount"));          // 为 bookCount 属性赋值
            book.setAuthor(rs.getString("author"));             // 为 author 属性赋值
            list.add(book);                                     // 将图书对象添加到集合中
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return list;                                                // 返回 list
}
```

由于存储过程 findAllBook 中没有定义参数，所以实例中通过调用“{call findAllBook()}”来调用存储过程。

注意　在通过 Connection 创建 CallableStatement 对象后，还需要调用其 executeQuery () 方法来执行存储过程，在调用后即可获取 ResultSet 对象得到查询结果集。

Step 4) 创建程序中的主页 index.jsp，在其中实例化 FindBook 对象，调用它的 find() 方法获取所有图书信息并显示在页面之中，其关键代码如下：

```
<jsp:useBean id="findBook" class="com.lyq.bean.FindBook"></jsp:useBean>
<table align="center" width="450" border="1">
    <tr>
        <td align="center" colspan="5">
            <h2> 所有图书信息 </h2>
        </td>
    </tr>
    <tr align="center">
        <td><b>ID</b></td>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            <td><b> 图书名称 </b></td>
            <td><b> 价格 </b></td>
            <td><b> 数量 </b></td>
            <td><b> 作者 </b></td>
    </tr>
            <%
                    // 获取图书信息集合
                            List<Book> list = findBook.findAll();
                            // 判断集合是否有效
                            if(list == null || list.size() < 1){
                                    out.print(" 没有数据！ ");
                            }else{
                                    // 遍历图书集合中的数据
                                    for(Book book : list){
            %>
                    <tr align="center">
                            <td><%=book.getId()%></td>
                            <td><%=book.getName()%></td>
                            <td><%=book.getPrice()%></td>
                            <td><%=book.getBookCount()%></td>
                            <td><%=book.getAuthor()%></td>
                    </tr>
            <%
                                    }
                    }
            %>
</table>
```

实例运行后打开 index.jsp 页面，程序执行数据库中定义的存储过程 findAllBook 查询图书信息，运行结果如图 10.26 所示。



图 10.26　运行结果

10.5　JDBC在Java Web中的应用

在 Java Web 开发中，JDBC 的应用十分广泛，Web 程序修改数据库的操作一般均通过 JDBC 实现。即使目前数据库方面的开源框架层出不穷，但其底层实现也离不开 JDBC API。

10.5.1　开发模式

在 Java Web 开发中使用 JDBC 应遵循 MVC（Model-View-Controller）的设计思想，使 Web 程序拥有一定的健壮性和可扩展性。

MVC 是一种程序设计理念，它将软件分成 3 层结构，即模型层、视图层与控制层。其

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

中，模型层泛指程序之中的业务逻辑，用于处理真正的业务操作；视图层指程序与用户相交互的界面，对用户呈现出的视图，但不包含业务逻辑；控制层分发处理用户各种请求，将指定的请求分配给指定的业务逻辑进行处理。

Java Web 中的 MVC 如图 10.27 所示。

客户端通过 JSP 页面与程序交互，对于数据的增加、删除、修改及查询请求由 Servlet 分发处理。如 Servlet 接收到添加数据请求，就会分发给添加数据的 JavaBean 对象，而对数据库的操作则通过 JDBC 封装的 JavaBean 实现。

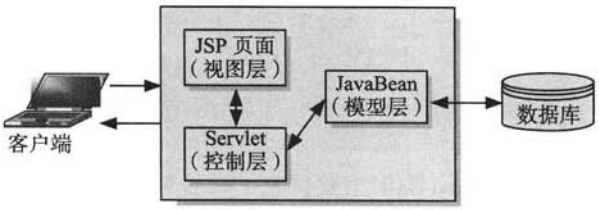


图 10.27　Java Web 中的 MVC

10.5.2　分页查询

当数据库中的信息量非常大时，不适合将所有数据显示在一个页面中，因为这样既为查看带来不便，又占用了程序及数据库的资源，为此需要分页查询。分页查询是 Java Web 开发之中十分常用的技术。

通过 JDBC 实现分页查询的方法有多种，而且不同的数据库机制也提供了不同的分页方式，两种典型的分页方法如下。

（1）通过 ResultSet 光标分页

ResultSet 对象是 JDBC API 中封装的查询结果集对象，通过它可以实现数据的分页显示。在该对象中有一个“光标”的概念，光标通过上下移动定位查询结果集中的行来获取数据。通过 ResultSet 移动光标，设置 ResultSet 对象中记录的起始和结束位置来实现数据的分页显示。

通过 ResultSet 光标实现分页的优点是在多种数据库上通用；缺点是占用大量资源，不适合于数据量大的情况。

（2）通过数据库机制分页

很多数据库提供了分页机制，如 SQL Server 中提供的“top”关键字和 MySQL 数据库中提供的“limit”关键字，它们都可以设置数据返回的记录数。

通过各种数据库提供的分页机制实现分页查询，其优点是减少数据库资源的开销，提高程序的性能；缺点是只针对某一种数据库。

说明　由于通过 ResultSet 的光标实现数据分页存在性能方面的缺陷，所以在实际开发之中，一般是采用数据库提供的分页机制来实现分页查询功能。

【例 10.10】　通过 MySQL 数据库提供的分页机制，实现商品信息的分页查询功能，将分页数据显示在 JSP 页面中。

实例位置：光盘\MR\源码\第 10 章\10.10

Step 1）创建名为 Product 的 JavaBean，用于封装商品信息，其关键代码如下：

```
public class Product {
    public static final int PAGE_SIZE = 2;          // 每页记录数
    private int id;                                 // 编号
    private String name;                            // 名称
    private double price;                           // 价格
    private int num;                                // 数量
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    private String unit;                                    // 单位
    public int getId() {
           return id;
    }
    public void setId(int id) {
           this.id = id;
    }
    // 省略部分 Setter 与 Getter 方法
}
```

在 Product 类中主要封装了商品对象的基本信息。除此之外，该类还定义了分页中每页的记录数。它是一个静态变量，可以直接引用。由于每页记录数不会经常修改，所以实例将其定义为 final 类型。

说明 在 Java 语言中如果定义了静态的 final 类型变量，通常情况下将这个变量大写。这是一种编写规范，能够很容易地区分其他类型的变量。

Step 2 创建名为 ProductDao 的类，主要用于封装商品数据库的相关操作。在 ProductDao 类中首先编写 getConnection() 方法，用于创建数据库连接 Connection 对象，其关键代码如下：

```
public class ProductDao {
 /**
  * 获取数据库连接
  * @return Connection 对象
  */
public Connection getConnection(){
       Connection conn = null;                              // 数据库连接
       try {
              Class.forName("com.mysql.jdbc.Driver");
                                                            // 加载数据库驱动，注册到驱动管理器
              String url = "jdbc:mysql://localhost:3306/db_database10";
                                                            // 数据库连接字符串
              String username = "root";                     // 数据库用户名
              String password = "111";                      // 数据库密码
              conn = DriverManager.getConnection(url,username,password);
                                                            // 创建 Connection 连接
       } catch (ClassNotFoundException e) {
              e.printStackTrace();
       } catch (SQLException e) {
              e.printStackTrace();
       }
       return conn;                                         // 返回数据库连接
}
```

技巧 由于 Connection 对象是每一个数据操作方法都要用到的对象，所以实例中通过封装 getConnection() 方法创建 Connection 对象，实现代码的重用。

Step 3 创建商品信息的分页查询方法 find()，其中包含一个 page 参数，用于传递要查询的页码，其关键代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
/**
 * 分页查询所有商品信息
 * @param page 页数
 * @return List<Product>
 */
public List<Product> find(int page){
 List<Product> list = new ArrayList<Product>();          // 创建 List
 Connection conn = getConnection();                      // 获取数据库连接
 String sql = "select * from tb_product order by id desc limit ?,?";
                                                         // 分页查询的 SQL 语句
 try {
       PreparedStatement ps = conn.prepareStatement(sql);
                                                         // 获取 PreparedStatement
       ps.setInt(1, (page - 1) * Product.PAGE_SIZE);     // 对 SQL 语句中的第 1 个参数赋值
       ps.setInt(2, Product.PAGE_SIZE);                  // 对 SQL 语句中的第 2 个参数赋值
          ResultSet rs = ps.executeQuery();              // 执行查询操作
       // 光标向后移动，并判断是否有效
       while(rs.next()){
            Product p = new Product();                   // 实例化 Product
            p.setId(rs.getInt("id"));                    // 为 id 属性赋值
            p.setName(rs.getString("name"));             // 为 name 属性赋值
            p.setNum(rs.getInt("num"));                  // 为 num 属性赋值
            p.setPrice(rs.getDouble("price"));           // 为 price 属性赋值
            p.setUnit(rs.getString("unit"));             // 为 unit 属性赋值
            list.add(p);                                 // 将 Product 添加到 List 集合中
       }
          rs.close();                                    // 关闭 ResultSet
          ps.close();                                    // 关闭 PreparedStatement
          conn.close();                                  // 关闭 Connection
   } catch (SQLException e) {
          e.printStackTrace();
   }
   return list;
}
```

find() 方法用于实现分页查询功能，此方法根据入口参数 page 传递的页码查询指定页码中的记录，主要通过 limit 关键字实现。

说明

MySQL 数据库提供的 limit 关键字能够控制查询数据结果集的起始位置及返回记录的数量，其使用方式如下：

```
limit arg1,arg2
```

- arg1：指定查询记录的起始位置。
- arg2：指定查询数据返回的记录数。

find() 方法主要应用 limit 关键字编写分页查询的 SQL 语句，其中 limit 关键字的两个参数通过 PreparedStatement 为其赋值，第 1 个参数为查询记录的起始位置，根据 find() 方法中的页码参数 page 可以计算，算法为 (page - 1) * Product.PAGE_SIZE；第 2 个参数为返回的记录数，即每一页所显示的记录数量，其值为 Product.PAGE_SIZE。

在为 SQL 语句传递了这两个参数后，执行 PreparedStatement 对象的 executeQuery() 方法即可获取指定页码中的结果集。实例中将所查询的商品信息封装为 Product 对象并放置到 List 集合中，最后将其返回。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

ProductDao 类主要用于封装商品信息的数据库操作，所以操作商品信息数据库的相关方法应定义在此类之中。在分页查询过程中还需要获取商品信息的总记录数，用于计算商品信息的总页数。此操作编写在 findCount() 方法中，其关键代码如下：

```
/**
 * 查询总记录数
 * @return 总记录数
 */
public int findCount(){
      int count = 0;                                          // 总记录数
      Connection conn = getConnection();                      // 获取数据库连接
      String sql = "select count(*) from tb_product";         // 查询总记录数 SQL 语句
      try {
            Statement stmt = conn.createStatement();          // 创建 Statement
            ResultSet rs = stmt.executeQuery(sql);            // 查询并获取 ResultSet
            if(rs.next()){                                    // 光标向后移动，并判断是否有效
                  count = rs.getInt(1);                       // 为总记录数赋值
            }
            rs.close();                                       // 关闭 ResultSet
            conn.close();                                     // 关闭 Connection
      } catch (SQLException e) {
            e.printStackTrace();
      }
      return count;                                           // 返回总记录数
 }
}
```

查询商品信息总记录数的 SQL 语句为“select count(*) from tb_product”，findCount() 方法主要通过执行这条 SQL 语句获取总记录数。

注意 获取查询结果需要调用 ResultSet 对象的 next() 方法向下移动光标，由于所获取的数据只是单一的数值，所以实例中通过 if(rs.next()) 调用，而没有使用 while 调用。

Step 4 创建名为 FindServlet 的类，此类是分页查询商品信息的 Servlet 对象。在该类中重写 doGet() 方法处理分页请求，其关键代码如下：

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
 int currPage = 1;                                        // 当前页码
 if(request.getParameter("page") != null){                // 判断传递页码是否有效
       currPage = Integer.parseInt(request.getParameter("page"));
                                                          // 对当前页码赋值
 }
 ProductDao dao = new ProductDao();                       // 实例化 ProductDao
 List<Product> list = dao.find(currPage);                 // 查询所有商品信息
 request.setAttribute("list", list);                      // 将 list 放置到 request 之中
 int pages ;                                              // 总页数
 int count = dao.findCount();                             // 查询总记录数
   if(count % Product.PAGE_SIZE == 0){                    // 计算总页数
       pages = count / Product.PAGE_SIZE;                 // 为总页数赋值
 }else{
       pages = count / Product.PAGE_SIZE + 1;             // 为总页数赋值
 }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
StringBuffer sb = new StringBuffer();                    // 实例化 StringBuffer
// 通过循环构建分页条
for(int i=1; i <= pages; i++){
      if(i == currPage){                                 // 判断是否为当前页
            sb.append("『" + i + "』");                   // 构建分页条
         }else{
            sb.append("<a href='FindServlet?page=" + i + "'>" + i + "</a>");
                                                         // 构建分页条
         }
         sb.append("  ");                                // 构建分页条
}
request.setAttribute("bar", sb.toString());              // 将分页条的字符串放置到 request 之中
request.getRequestDispatcher("product_list.jsp").forward(request, response);
                                                         // 转发到 product_list.jsp 页面
}
```

FindServlet 类的 doGet() 方法主要获取分页查询结果集并构造分页条对象，其中获取分页查询结果通过调用 ProductDao 类的 find() 方法，并传递所要查询的页码实现；分页条对象是 JSP 页面中的分页条，用于显示商品信息的页码，主要通过创建页码的超链接，然后组合字符串构造。

技巧 分页条在 JSP 页面中是动态内容，每次查看新页面都要重新构造。所以实例中将分页的构造放置到 Servlet 之中，以简化 JSP 页面的代码。

在构建分页条时需要计算商品信息的总页码，其值通过总记录数与每页记录数计算得出。计算得出总页码后，实例中通过 StringBuffer 组合字符串构建分页条。

技巧 如果一个字符串经常发生变化，应该使用 StringBuffer 操作字符。因为在 JVM 中每次创建一个新的字符串都要分配一个字符串空间。而 StringBuffer 则是字符串的缓冲区，所以在经常修改字符串的情况下，StringBuffer 的性能更高。

在获取查询结果集 List 与分页条后，FindServlet 分别将这两个对象放置到 request 之中，将请求转发到 product_list.jsp 页面显示。

Step 5) 创建 product_list.jsp 页面，此页面通过获取查询结果集 List 与分页条来分页显示商品信息数据，其关键代码如下：

```
<table align="center" width="450" border="1">
   <tr>
         <td align="center" colspan="5">
               <h2> 所有商品信息 </h2>
         </td>
   </tr>
   <tr align="center">
         <td><b>ID</b></td>
         <td><b> 商品名称 </b></td>
         <td><b> 价格 </b></td>
         <td><b> 数量 </b></td>
         <td><b> 单位 </b></td>
   </tr>
   <%
         List<Product> list = (List<Product>)request.getAttribute("list");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            for(Product p : list){
    %>
    <tr align="center">
            <td><%=p.getId()%></td>
            <td><%=p.getName()%></td>
            <td><%=p.getPrice()%></td>
            <td><%=p.getNum()%></td>
            <td><%=p.getUnit()%></td>
    </tr>
    <%
            }
    %>
    <tr>
            <td align="center" colspan="5">
                    <%=request.getAttribute("bar")%>
            </td>
    </tr>
</table>
```

查询结果集 List 与分页条均从 request 对象中获取，其中结果集 List 通过 for 循环遍历并将每一条商品信息输出到页面中，分页条输出到商品信息下方。

Step 6) 编写程序中的主页面 index.jsp，在其中编写分页查询商品信息的超链接指向 FindServlet，其关键代码如下：

```
<body>
    <a href="FindServlet">查看所有商品信息 </a>
</body>
```

运行本实例，将分页显示商品信息，如图 10.28 所示，在该图中可以到商品信息的分页条，本例中所有商品信息分 3 页显示。单击分页条中超链接可以查看指定页面的商品信息，例如，单击 2 超链接，将显示第 2 页数据，如图 10.29 所示。



图 10.28 分页商品信息



图 10.29 第 2 页数据

10.6 上机实践

10.6.1 批量删除数据

➤➤➤ 题目描述

在对数据进行删除时有很多种方法。本上机实践尝试通过复选框实现数据的批量删除功能，可选择一条或多条数据进行删除处理，运行结果如图 10.30 所示。通过选择每条数据后面

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

的复选框进行多条数据的选择，将要删除的数据选中后，单击“删除”按钮，即可将所有被选中的数据删除。



图 10.30　批量删除图书信息

➤➤➤ 技术指导

在实现本实例时，首先在显示图书信息列表的页面中，通过复选框设置可以选择多条要删除的图书信息，然后单击“删除”超链接。在 Servlet 中，编写预编译执行的 SQL 语句，并通过 for 循环将获取到的复选框的值（即要删除图书的编号）设置为 SQL 语句的参数，再添加批处理命令，最后执行批处理操作。本上机实践的关键代码如下：

```
Connection conn = DriverManager.getConnection(url, username,password);
                                                    // 创建 Connection 连接
String sql = "DELETE FROM tb_book WHERE id=?";      // 更新 SQL 语句
PreparedStatement ps = conn.prepareStatement(sql);  // 获取 PreparedStatement
String ID[]=request.getParameterValues("delid");    // 获取要删除的图书编号
if (ID.length>0){
   for(int i=0;i<ID.length;i++){
          ps.setInt(1,Integer.parseInt(ID[i]));     // 对 SQL 语句中的第 1 个参数赋值
          ps.addBatch();                            // 添加批处理命令
   }
}
ps.executeBatch();                                  // 执行批处理操作
```

10.6.2　网站用户注册

➤➤➤ 题目描述

现实中很多网站实行会员制，一些网站内容需要用户注册成为会员后才可以浏览。本上机实践尝试完成一个网站的会员注册模块，运行结果如图 10.31 所示。



图 10.31　运行结果

➤➤➤ 技术指导

完成本实例的步骤如下。

Step 1) 新建动态 Web 项目，将 MySQL 数据库驱动包复制到项目中的 WEB-INF/lib 目录下，新建用户注册表单页 index.jsp。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Step 2) 新建名为 registerdo.jsp 的处理页，用于获取用户注册的表单信息并处理。在该页中首先判断用户名是否存在，如果存在，则把信息封装到 User 类中并调用 UserDao 类的方法将信息保存到数据库；否则提示错误信息。

Step 3) 创建成功页 ok.jsp 和错误页 error.jsp，用于输出提示信息。

Step 4) 创建名为 User 的类，用于封装用户的注册信息。

Step 5) 创建数据库连接类 DBCon。

Step 6) 创建操作数据库的类 UserDao，其中包含一个保存用户注册信息的方法和一个判断用户名是否存在的方法。

10.6.3 插入用户登录日志信息

➤➤➤ 题目描述

在网站中，通常会提供一个用户登录页面，该页面用于验证用户是否有权限登录此网站。为了实时掌握哪些用户何时访问过网站，在用户登录的同时也会将用户的登录信息记录到数据库中。本上机实践尝试完成插入用户登录日志信息的程序，运行结果如图 10.32 和图 10.33 所示。



图 10.32 用户登录页面

id	name	pwd	loginTime	info
1	无语	111	2012-07-23 16:52:05	用户登录

图 10.33 保存到数据库中的日志信息

➤➤➤ 技术指导

在实现本实例时，首先通过表单提交用户登录信息，然后在 Servlet 中获取提交的用户名和密码，再获取当前的系统时间，并将获取到的这些信息组合成插入数据的 SQL 语句，最后执行该 SQL 语句完成登录日志信息的插入。

本上机实践的关键代码如下：

```
String userName=request.getParameter("tname");
String passWord=request.getParameter("tpassword");
SimpleDateFormat date_time = new SimpleDateFormat("yyyy-MM-dd HH;mm;ss");
String datetime = date_time.format(new Date());
String info="用户登录";
String sql="insert into tb_login (name,pwd,loginTime,info) values('"+userName+
          "','"+passWord+"','"+datetime+"','"+info+"')";
dao.executeUpdate(sql);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

10.7　小结

本章首先介绍了 JDBC 技术及 JDBC 连接数据的过程，然后介绍了 JDBC API 中的常用对象和通过 JDBC 操作数据库的方法，最后介绍了 JDBC 在 Java Web 中的应用。读者需要重点掌握通过 JDBC API 实现数据的增、删、改、查方法，要理解 MVC 的设计思想，掌握 JDBC 在 Java Web 中的开发模式及数据的分页查询操作。

10.8　习题

选择题

1. MySQL 使用的默认端口是（　　）。
 A. 1433　　B. 8080　　C. 3306
2. 下面哪个接口用于执行静态的 SQL 语句？（　　）
 A. PreparedStatement　　B. Statement　　C. Connection
3. 下面（　　）接口用于保存 JDBC 执行查询时返回的结果集，该结果集与数据库表字段相对应。
 A. ResultSet　　B. Connection　　C. Statement

填空题

1. JDBC 是_____（Application Programming Interface，应用程序编程接口），定义在 JDK 的 API 之中。
2. 执行动态 SQL 语句的接口是_____。
3. Class 的 forName() 方法的作用是_____。
4. 应用 PreparedStatement 接口中的 SQL 语句可以使用占位符_____来代替其参数，然后通过_____方法为 SQL 语句的参数赋值。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第三篇
高级篇

本篇主要内容：

- ❑ 第 11 章　在互联网中操作文件资源
- ❑ 第 12 章　利用 JFreeChart 实现动态图表
- ❑ 第 13 章　在 JSP 中操作 XML
- ❑ 第 14 章　完美体验 Ajax 技术
- ❑ 第 15 章　MVC 架构实现者——Struts 2 框架
- ❑ 第 16 章　数据库持久化利器——Hibernate 技术
- ❑ 第 17 章　简化企业开发瑰宝——Spring 技术

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 11 章　在互联网中操作文件资源

视频讲解：83 分钟

在开发网站时，经常需要实现操作文件资源的功能，常用的文件操作包括文件上传与下载。在 Java Web 中，实现文件上传通常有两种方法一种是使用 Servlet 3.0 的新特性来实现，另一种是使用 Common-FileUpload 组件实现；实现文件下载通常使用流实现。本章将详细如何介绍在 Java Web 中实现文件下传与下载。

通过学习本章，读者可以达到以下学习目的：

- 掌握使用 Servlet 3.0 的新特性实现文件上传；
- 掌握如何使用 Commons-FileUpload 组件上传文件；
- 掌握如何使用流实现文件下载。

11.1　使用Servlet 3.0的新特性实现文件上传

在 Servlet 3.0 出现之前，不使用第三方组件实现文件上传是一件很麻烦的事情，而 Servlet 3.0 出现以后就解决了这一问题。使用 Servlet 3.0 可以十分方便地实现文件的上传。实现文件上传需要以下两项内容：

- 添加 @MultipartConfig 注解。
- 从 request 对象中获取 Part 文件对象。

@MultipartConfig 注解需要标注在 @WebServlet 注释之上，具有如表 11.1 所示的常用属性。

表 11.1　@MultipartConfig 注释的常用属性

属 性 名	类　型	是否可选	描　述
fileSizeThreshold	int	是	当数据量大于该值时，内容将被写入文件
location	String	是	存放临时生成的文件地址
maxFileSize	long	是	允许上传的文件最大值。默认值为 -1，表示没有限制
maxRequestSize	long	是	针对该 multipart/form-data 请求的最大数量，默认值为 -1，表示没有限制

除了要配置 @MultipartConfig 注解之外，还需要使用两个重要的方法，getPart() 方法与 getParts() 方法。下面对这两个方法进行详细的介绍。

- Part getPart(String name)。
- Collection<Part> getParts()。

getPart() 方法的 name 参数表示请求的 name 文件。getParts() 方法可获取请求中的所有文件。上传文件用 javax.servlet.http.Part 对象来表示。Part 接口提供了处理文件的简易方法，比如 write() 和 delete() 方法等。

【例 11.1】 应用 Servlet 实现文件上传。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

实例位置：光盘\MR\源码\第11章\11.1

Step 1) 编写具有上传文件组件的 JSP 页面，在该页面中还包含有“上传”按钮，代码如下：

```
<form action="UploadServlet" enctype="multipart/form-data" method ="post" >
    选择文件<input type="file" name="file1" id= "file1"/>
    <input type="submit" name="upload" value=" 上传 " />
 </form>
```

Step 2) 编写处理上传文件的 Servlet，在该 Servlet 中对上传文件进行控制，代码如下：

```
@WebServlet("/UploadServlet")
@MultipartConfig(location = "d:\temp")
public class UploadServlet extends HttpServlet {
 private static final long serialVersionUID = 1L;
 protected void doPost(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
       response.setContentType("text/html;charset=UTF-8");
       PrintWriter out = response.getWriter();
       String path = this.getServletContext().getRealPath("/");
                                                          //获取服务器地址
       Part p = request.getPart("file1");                 //获取用户选择的上传文件
       if (p.getContentType().contains("image")) {        // 仅处理上传的图像文件
            ApplicationPart ap = (ApplicationPart) p;
            String fname1 = ap.getFilename();             //获取上传文件名
            int path_idx = fname1.lastIndexOf("\\") + 1;  //对上传文件名进行截取
            String fname2 = fname1.substring(path_idx, fname1.length());
            p.write(path + "/upload/" + fname2);
                                          // 写入 web 项目根路径下的 upload 文件夹中
            out.write(" 文件上传成功 ");
       }
       else{
            out.write(" 请选择图片文件！！！ ");
       }
  }
}
```

运行本程序，选择上传文件后，如果上传文件是图片文件，单击“上传”按钮后，即可实现文件上传，如图 11.1 所示。

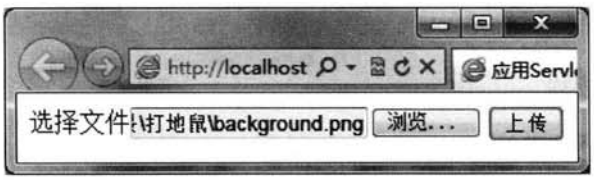


图 11.1　上传文件

11.2　使用Commons-FileUpload组件实现文件上传

Commons-FileUpload 组件是 Apache 组织 jakarta-commons 项目组下的一个小项目，该组

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

件可以方便地将 multipart/form-data 类型请求中的各种表单域解析出来，并实现一个或多个文件的上传，同时也可以限制上传文件的大小等内容。在使用 Commons-FileUpload 组件时，需要先下载该组件。该组件可以到 http://commons.apache.org/fileupload/ 网站下载。

说明 Commons-FileUpload 组件需要 commons-io 包的支持，所以在下载 Commons-FileUpload 组件时，还需要一起下载 commons-io 组件。

11.2.1 添加表单及表单元素

在上传文件页面中，要添加用于上传文件的表单及表单元素。在该表单中，需要通过文件域指定要上传的文件。在表单中添加文件域的语法格式如下：

```
<input name="file" type="file" size="尺寸 ">
```

- name 属性：用于指定文件域的名称。
- type 属性：用于指定标记的类型，这里设置为 file，表示文件域。
- size 属性：用于指定文件域中文本框的长度。

例如，在表单中添加一个名称为 file 的文件，可以使用下面的代码：

```
<input name="file" type="file" size="35">
```

注意 在实现文件上传时，必须将 form 表单的 enctype 属性设置为“multipart/form-data”，否则将不能上传文件。

11.2.2 创建上传对象

在应用 Commons-FileUpload 组件实现文件上传时，需要创建一个工厂对象，并根据该工厂对象创建一个新的文件上传对象，具体代码如下：

```
DiskFileItemFactory factory = new DiskFileItemFactory();
                                                    //基于磁盘文件项目创建一个工厂对象
ServletFileUpload upload = new ServletFileUpload(factory);
                                                    //创建一个新的文件上传对象
```

在使用上面的两行代码时，需要导入相应的类，具体的代码如下：

```
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;
```

11.2.3 解析上传请求

创建一个文件上传对象后，就可以应用这个对象解析上传请求。在解析上传请求时，首先要获取全部的表单项，这可以通过文件上传对象的 parseRequest() 方法来实现。parseRequest() 方法原型如下：

```
public List parseRequest(HttpServletRequst request) throws FileUploadException
```

request：HttpServletRequest 对象。

例如，应用该方法获取全部表单项，并保存到 items 中的具体代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
List items = upload.parseRequest(request);                    // 获取全部的表单项
```

通过 parseRequest() 方法获取的全部表单项将保存到 List 集合中，并且保存到 List 集合中的表单项，不管是文件域还是普通表单域，都将当成 FileItem 对象处理。在进行文件上传时，可以通过 FileItem 对象的 isFormField() 方法判断表单项是文件域还是普通表单域。如果该方法的返回值为 true，则表示是一个普通表单域，否则是一个文件域。isFormField() 方法的原型如下：

```
public boolean isFormField()
```

例如，应用 isFormField() 方法判断文件域的具体代码如下：

```
if (!item.isFormField()) {                                    // 判断是否为文件域
    …       // 此处省略了部分代码
}
```

在实现文件上传时，还需要获取上传文件的文件名，这可以通过 FileItem 类的 getName() 方法实现，getName() 方法的原型如下：

```
public String getName();
```

注意　getName() 方法仅当该表单域是文件域时，才有效。

例如，通过 getName() 方法获取上传文件的文件名的具体代码如下：

```
String fileName=item.getName();                               // 获取文件名
```

在上传文件时，还可以通过 getSize() 方法获取上传文件大小。getSize() 方法的原型如下：

```
public long getSize()
```

例如，通过 getSize() 方法获取上传文件大小的具体代码如下：

```
long upFileSize=item.getSize();                               // 获取上传文件的大小
```

在上传文件时，还可以通过 getContentType() 方法获取上传文件的类型。getContentType() 方法的原型如下：

```
java.lang.String getContentType()
```

例如，通过 getContentType() 方法获取上传文件类型的具体代码如下：

```
String type=item.getContentType();                            // 获取文件类型
```

【例 11.2】　应用 Commons-FileUpload 组件将文件上传到服务器。

实例位置：光盘 \MR\ 源码 \ 第 11 章 \11.2

Step 1　创建 index.jsp 页面，要在其中包含文件上传表单项，实现文件上传操作时，需要将 form 表单的 enctype 属性值设置为 multipart/form-data。index.jsp 页面的关键代码如下：

```
<!-- 定义表单 -->
<form action="UploadServlet" method="post" enctype="multipart/form-data" name=
"form1" id="form1"
      onsubmit="return validate()">
      <ul>
          <li>请选择要上传的附件：</li>
          <li>上传文件：  <input type="file" name="file" /> <!-- 文件上传组件 --></li>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            <li><input type="submit" name="Submit" value="上传" />
                <input type="reset" name="Submit2" value="重置" /></li>
        </ul>
        <%
            if (request.getAttribute("result") != null) {
                                        //判断保存在 request 范围内的对象是否为空
                out.println("<script >alert('" + request.getAttribute("result")
                            + "');</script>");              //页面显示提示信息
            }
        %>
</form>
```

Step 2 当用户单击“上传”按钮时，系统将提交 URL 地址为 UploadServlet 的 Servlet，在该 Servlet 中处理文件上传请求。关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
 String adjunctname ;
 String fileDir = request.getRealPath("upload/");          //指定上传文件的保存地址
 String message = "文件上传成功";
 String address = "";
 if(ServletFileUpload.isMultipartContent(request)){        //判断是否是上传文件
       DiskFileItemFactory factory = new DiskFileItemFactory();
       factory.setSizeThreshold(20*1024);                  //设置内存中允许存储的字节数
       factory.setRepository(factory.getRepository());     //设置存放临时文件的目录
       ServletFileUpload upload = new ServletFileUpload(factory);
                                                           //创建新的上传文件句柄
       int size = 2*1024*1024;                             //指定上传文件的大小
       List formlists = null;                              //创建保存上传文件的集合对象
       try {
            formlists = upload.parseRequest(request);      //获取上传文件集合
       } catch (FileUploadException e) {
            e.printStackTrace();
       }
       Iterator iter = formlists.iterator();               //获取上传文件迭代器
       while(iter.hasNext()){
            FileItem formitem = (FileItem)iter.next();     //获取每个上传文件
            if(!formitem.isFormField()){                   //忽略不是上传文件的表单域
                  String name = formitem.getName();        //获取上传文件的名称
                  if(formitem.getSize()>size){ //如果上传文件大于规定的上传文件的大小
                        message = "您上传的文件太大，请选择不超过 2MB 的文件";
                        break;                             //退出程序
                    }
                  String adjunctsize = new Long(formitem.getSize()).toString();
                                                           //获取上传文件的大小
                  if((name == null) ||(name.equals(""))&&(adjunctsize.equals("0")))
                                                           //如果上传文件为空
                        continue;                          //重新开始循环
                  adjunctname = name.substring(name.lastIndexOf("\\")+1,name.length());
                  address = fileDir+"\\"+adjunctname;      //创建上传文件的保存地址
                  File saveFile = new File(address);       //根据文件保存地址，创建文件
                  try {
                       formitem.write(saveFile);           //向文件写数据
                  } catch (Exception e) {
                       e.printStackTrace();
                  }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            }
        }
    }
    request.setAttribute("result", message);          //将提示信息保存在 request 对象中
    RequestDispatcher requestDispatcher = request
            .getRequestDispatcher("index.jsp");       //设置相应返回地址
    requestDispatcher.forward(request, response);
}
```

运行本实例，单击“浏览”按钮，选择要上传的文件，注意要上传的文件不能大于 2MB，如图 11.2 所示，单击“上传”按钮即可将该文件上传到服务器的指定文件夹中。



图 11.2　上传文件到服务器上

11.3　实现文件下载

对于文件下载，大家也不陌生，相信多数人都有过从网络中下载 MP3 歌曲或电影的经历。随着网络的不断发展，文件下载功能远不止下载歌曲和电影了。现在很多网站都提供下载各类资源的功能。因此，有必要学习如何实现文件下载。

由于实现文件下载功能比较简单，通常情况下，不使用第三方组件实现，而是直接使用 Java 语言的输入 / 输出流实现。

在实现文件下载时，通常需要文件类（File）、文件字节输入流类（FileInputStream）和字节输出流类（OutputStream）。

11.3.1　文件类

java.io 包中的 File 类封装了系统的文件或目录的相关信息。例如，文件类型、文件路径等。File 类提供了以下两种创建 File 对象的方法。

方法 1：通过指定的文件路径字符串来创建 File 对象，语法如下。

```
File(String path);
```

例如，创建文件 ps.jpg（位于 E:\11\upload 目录下）所对应的 File 对象，可以使用下面的代码：

```
File file = new File("E:\\11\upload\ps.jpg");
```

方法 2：通过指定父路径与子路径字符串来创建 File 对象，语法如下。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
File(String parent,String child);
```

其中，parent 指定文件的父路径，child 指定文件的子路径。

例如，创建文件 ps.jpg（位于 E:\11\upload 目录下）所对应的 File 对象，可以使用下面的代码：

```
File file = new File("E:\\11\upload","ps.jpg");
```

技巧 在进行文件下载时，创建要下载文件所对应的文件对象时，通常应用 HttpServletRequest 对象的 getRealPath() 方法获取文件所在的绝对路径。例如，在当前 Web 应用中，要创建 upload 目录下的指定文件所对应的 File 对象，可以使用下面的代码。

```
File file = new File(request.getRealPath("/upload"), "ps.jpg");
                                                    // 创建要下载文件所对应的文件对象
```

11.3.2 文件字节输入流

文件字节输入流 FileInputStream 类，继承自 InputStream，可以从指定的文件中读取字节数据。可通过指定的文件对象来构建 FileInputStream 对象，或根据指定的文件名称和路径来构建 FileInputStream 类对象。

语法 1：根据指定的文件对象创建 FileInputStream 对象。

```
FileInputStream(File file)
```

例如，创建一个文件对象，再应用该对象的一个 FileInputStream 对象，可以使用下面的代码：

```
File file = new File("E:\\11\upload","ps.jpg");
FileInputStream in = new FileInputStream(file);
```

语法 2：根据文件名称与文件路径构建 FileInputStream 对象。

```
FileInputStream(String path)
```

通过 FileInputStream 类的 read() 方法，可以读取指定文件的内容。

例如，读取 E:\text\temp.txt 文件的内容可以使用下面的代码：

```
File file = new File("E:\\text\temp.txt");                    //创建文件对象
try {
        FileInputStream fis = new FileInputStream(file);      //创建文件字节输入流
        int length;
        while((length = fis.read())!=-1){                     //循环读取文件内容
        System.out.println((char)length);
   }
   fis.close();                                               //关闭文件字节输入流
} catch (Exception e) {
        e.printStackTrace();
}
```

说明 上面的方法只有读取的文件内容是英文或数字时，才能正常显示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

11.3.3　字节输出流

字节输出流 OutputStream 类中，定义了多种操作输出流的方法。通过字节输出流可以将数据以字节为单位输出并保存到指定文件中。字节输出流对象可以通过 HttpServletResponse 类的 getOutputStream() 方法获取。例如，在 Servlet 中，获取字节输出流对象，可以使用下面的代码：

```
OutputStream os = response.getOutputStream(); // 创建字节输出流对象
```

创建字节输出流对象后，就可以通过其 write() 方法实现向文件中写入数据。

例如，将文件字节输入流读取的内容通过字节输出流写入到指定文件中，可以使用下面的代码：

```
FileInputStream in = new FileInputStream(file);          // 创建文件字节输入流
OutputStream os = response.getOutputStream();            // 创建输出流对象
int data = 0;
while ((data = in.read()) != -1) {                       // 循环读取文件
   os.write(data);                                       // 向指定目录中写文件
}
```

【例 11.3】 采用输出流方式实现文件下载。

实例位置：光盘 \MR\ 源码 \ 第 11 章 \11.3

Step 1) 创建 index.jps 页面，在其中为用户提供 MP3 文件下载超链接，关键代码如下：

```
   <table style="margin-top:10px;" width="500" border="1" align="center"
cellpadding="0" cellspacing="0"
bordercolor="#FFFFFF" bordercolorlight="#FFFFFF" bordercolordark="#FFCCCC">
  <tr>
    <td width="29%" height="27" align="center"> 标题 </td>
    <td width="14%" height="27" align="center"> 上传人 </td>
    <td width="23%" height="27" align="center"> 上传时间 </td>
    <td width="6%" height="27" align="center"> 下载 </td>
    </tr>
  <tr>
    <td height="29" align="center"> 新歌 </td>
    <td align="center"> 小雨 </td>
    <td align="center">2012 年 7 月 24 日 </td>
    <td align="center">
      <a href="DownServlet?path=<%=getServletContext().getRealPath(" 新歌 .mp3")
      %>"> 下载 </a>
    </td>
    </tr>
</table>
```

Step 2) 当单击“下载”超链接时，系统将请求提交到名称为 DownServlet 的 Servlet。在该 Servlet 中，首先获取下载文件的地址，并根据该地址创建文件字节输入流，再通过该流读取下载文件内容，最后将读取的内容通过输出流写到目标文件中。具体代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
   String path=request.getParameter("path");              // 获取上传文件的路径
   path=new String(path.getBytes("ISO-8859-1"),"UTF-8");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    File file = new File(path);                                    //根据该路径创建文件对象
    InputStream in = new FileInputStream(file);                    //创建文件字节输入流
    OutputStream os = response.getOutputStream();                  //创建输出流对象
    response.addHeader("Content-Disposition", "attachment;filename="
                  + new String(file.getName().getBytes("GBK"),"ISO-8859-1"));
                                                                   //设置应答头信息
    response.addHeader("Content-Length", file.length() + "");
    response.setCharacterEncoding("UTF-8");
    response.setContentType("application/octet-stream");
    int data = 0;
    while ((data = in.read()) != -1) {                             //循环读取文件
           os.write(data);                                         //向指定目录中写文件
    }
    os.close();                                                    //关闭流
    in.close();
}
```

运行本实例，将显示如图 11.3 所示的运行结果。



图 11.3　采用输出流方式实现文件下载

11.4　上机实践

11.4.1　获取表单的元素值

➤➤➤ 题目描述

在实现文件上传时，由于将表单的属性 enctype 设置为 multipart/form-data，所以通过 request 对象的 getParameter() 方法无法获取，为此需要特别处理。本上机实践尝试应用 Commons-FileUpload 组件获取其他表单元素，实例运行结果如图 11.4 所示。

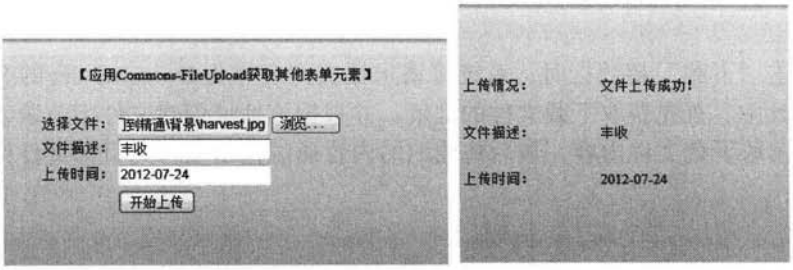


图 11.4　上传文件并获取其他表单元素的值

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

➤➤➤ 技术指导

commons-fileupload 组件在处理表单时，需要使用 ServletFileUpload 类的 parseRequest (request) 方法获取上传文件的 List 集合，然后使用 isFormFile() 方法判断是普通的表单属性还是一个文件。如果是普通的表单属性，可以通过以下方法获取表单元素（返回值为 String 类型）：

```
fileItem.getFieldName();
```

获取表单元素的值可以应用 FileItem 对象的 getString() 方法。在使用 getString() 方法获取表单元素值时，为了避免出现乱码，可以设置值的编码格式，例如：

```
String user="";
if(fileItem.getFieldName().equals("userName")){
        user = fileItem.getString("UTF-8");
}
```

11.4.2　限制文件上传的类型

➤➤➤ 题目描述

在实现文件上传时，有时考虑到网站的安全，需要限制上传文件的类型，对于一些可能对网站构成威胁的文件将禁止上传。本上机实践尝试应用 commons-fileupload 组件实现限制上传文件的上传类型。运行结果如图 11.5 所示。当上传文件类型为 *.exe、*.bat 或 *.jsp 时，将弹出禁止上传的提示信息。



图 11.5　限制文件的上传类型

➤➤➤ 技术指导

本实例主要应用 commons-io 组件的 org.apache.commons.io.filefilter 包中的 SuffixFileFilter 类，该类主要用于设置过滤文件的后缀名，也就是设置过滤文件类型。SuffixFileFilter 类实现自 java.io.FileFilter 接口，该类的方法说明如表 11.2 所示。

表 11.2　SuffixFileFilter 类的常用方法说明

方　法	说　明
public SuffixFileFilter(String suffix)	构造方法，参数 suffix 表示要过滤的文件后缀字符串
public SuffixFileFilter(String[] suffixes)	构造方法，参数 suffixes 表示要过滤的文件后缀字符串数组

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（续）

方　法	说　明
public SuffixFileFilter(List suffixes)	构造方法，参数 suffixes 表示要过滤的文件后缀字符串集合
public boolean accept(File file)	过滤方法，参数 file 表示要进行过滤的文件对象。如果 file 对象的文件类型与构造方法中指定的字符串 suffix、字符串数组 suffixes 或集合 suffixes 表示的后缀相同，返回 true，否则返回 false

本上机实践的关键代码如下：

```
String[] suffixs =new String[]{".exe",".bat",".jsp"};     //限制上传的文件类型
SuffixFileFilter filter = new SuffixFileFilter(suffixs);
for(FileItem fileItem:files){                                    //遍历上传文件集合
   if(!fileItem.isFormField())     {        //忽略其他不是文件域的所有表单信息
          String filePath = fileItem.getName();                  //获取文件全路径名
          String fileName="";
          int startIndex = filePath.lastIndexOf("\\");
          if(startIndex!=-1){                                    //对文件名进行截取
                 fileName = filePath.substring(startIndex+1);
          }else{
                 fileName=filePath;
          }
          if(fileItem.getSize()>maxSize){                        //限制文件大小
                 message = "上传文件不得超过5MB！";
                 break;
          }
          if((fileName == null) ||(fileName.equals(""))&&(fileItem.getSize()==0))
                 continue;
          File file = new File(uploadPath,fileName);             //在上传路径创建文件对象
          boolean res = filter.accept(file);
          if(res){
                 message = "禁止上传 *.exe、*.jsp、*.bat 文件！";
                 break;
          }else{
                 fileItem.write(file);                           //向文件写数据
          }
   }
}
```

11.4.3　批量文件上传

➤➤➤ 题目描述

在 Web 项目开发时，有时需要实现批量文件上传功能。使用 Commons-FileUpload 组件实现批量上传文件也比较简单。本上机实践尝试应用 Commons-FileUpload 组件实现批量上传文件，实例运行结果如图 11.6 所示。

➤➤➤ 技术指导

应用 Commons-FileUpload 组件实现批量上传文件与单个文件上传类似，所不同的是，前台表单中需要提供多个文件域，用于指定要上传的文件，同时在处理上传请求时，分别对多个文件域内容进行获取并保存。本上机实践的关键代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 11.6　批量文件上传

```
List<FileItem> files = upload.parseRequest(request);      //从请求中得到所有上传域列表
for(FileItem fileItem:files){                              //遍历上传文件集合
   if(!fileItem.isFormField()){                          //忽略其他不是文件域的所有表单信息
          String name = fileItem.getName();
          String user="";
          if(fileItem.getFieldName().equals("userName"))
                 user = fileItem.getString("GBK");
          if(fileItem.getSize()>upload.getFileSizeMax()){ //限制文件大小
           message = "上传文件不得超过5MB！";
           break;
       }
          if((name == null) ||(name.equals(""))&&(fileItem.getSize()==0))
                 continue;
          File file = new File(uploadPath,name);           //在上传路径创建文件对象
          fileItem.write(file);                            //向文件写数据
   }
}
```

11.5　小结

本章主要介绍了在 Java Web 开发中常用的文件上传与下载的方法。首先介绍的是使用 Servlet 3.0 的新特性实现文件上传，该方法比较简单，如果你的服务器支持 Servlet 3.0，则使用该方法实现文件上传是个不错的选择；然后介绍了使用 Commons-FileUpload 组件实现文件上传，Commons-FileUpload 组件是一个比较成熟的组件，使用它实现文件上传也比较方便；最后介绍了通过流实现文件下载，文件下载也比较重要，需要重点掌握。

11.6　习题

选择题

1．在表单中添加文件域的语法格式所包含的属性不包括（　　）。

A. name　　B. type　　C. nice

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2. 解析上传请求中 getName() 方法时，仅当该表单域是（　　）时才有效。
 A. 文件域　　B. 文本域　　C. 文本框
3. 通过 FileInputStream 类的（　　）方法，可以读取指定文件的内容。
 A. get()　　B. read()　　C. write()
4. 字节输出流中写文件的方法是（　　）。
 A. out()　　B. put()　　C. write()
5. 文件上传的三个步骤中，判断文件类型是在哪一步中？（　　）
 A. 添加表单及表单元素　　B. 创建上传对象　　C. 解析上传请求

填空题

1. 使用 Servlet 3.0 的新特性实现文件上传时需要设置_____和_____两项内容。
2. 在实现文件上传时，必须将 form 表单的 enctype 属性设置为_____，否则将不能上传文件。
3. 获取全部的表单项，可以通过文件上传对象的_____方法来实现。
4. 字节输出流对象可以通过 HttpServletResponse 类的_____方法获取。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 12 章　利用 JFreeChart 实现动态图表

视频讲解：39 分钟

以图形报表的形式对数据进行统计分析，其显示结果非常直观、大方，使查看者能够一目了然。然而图表的制作非常烦琐，而且当数据发生变化时，需要对其进行重新绘制。在 Java Web 开发中，JFreeChart 组件为图形报表技术提供了解决方案。

JFreeChart 组件用于绘制动态图表，其强大的功能、优秀的制图效果以及便捷的操作方法，在 Java Web 中得到一致的认可。本章将介绍 JFreeChart 组件的使用方法。

通过学习本章，读者可以达到以下学习目的：

- 了解动态图表；
- 掌握 JFreeChart 组件的制图方法；
- 理解制图对象；
- 理解数据集合；
- 掌握常用绘图区对象类型及关系；
- 掌握坐标轴对象类型及关系；
- 掌握图片渲染对象；
- 了解 JFreeChart 组件的内置 JDBC。

12.1　JFreeChart简介

JFreeChart 是 Java 中开源的制图组件，主要用于生成各种动态图表，在 Java 的图形报表技术中，JFreeChart 组件提供了方便、快捷、灵活的制图方法。

12.1.1　认识 JFreeChart 组件

JFreeChart 是 Java 领域中一个强大的图形报表组件，它为 Java 的图形报表技术提供了解决方案。在 Java 项目的应用中，JFreeChart 组件几乎可以满足目前图形报表的所有需求，其功能十分强大。

JFreeChart 组件可以生成各种各样的图形报表，如常用的柱形图、区域图、饼形图、折线图、时序图、甘特图等，而对于同一种类型的图表，JFreeChart 组件还提供了不同的表现方式。这些效果可以在下载 JFreeChart 组件后，通过运行 JFreeChart 提供的演示程序 jfreechart-1.0.14-demo.jar 文件进行查看，其运行后的效果如图 12.1 所示。

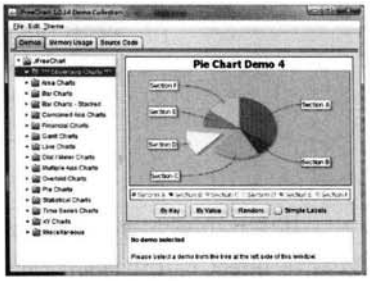


图 12.1　JFreeChart 演示程序

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

12.1.2 JFreeChart 的下载与使用

JFreeChart 是开源站点 SourceForge.net 上的一个 Java 项目，它是开放源代码的图形报表组件，其主页为“http://www.jfree.org/jfreechart”，在此页面可以下载 JFreeChart 组件的最新版本。打开主页面，如图 12.2 所示，单击页面中的 DOWNLOAD 导航链接将进入下载页面，选择所要下载的产品 JFreeChart 即可进行下载。在本书编写时它的最新版本为 1.0.14 版本，本章内容将以此版本为例进行讲解。



图 12.2 JFreeChart 主页

下载成功后将得到一个名称为“jfreechart-1.0.14.zip”的压缩包，此压缩包中包含 JFreeChart 组件的源码、示例、支持类库等文件，将其解压缩后的文件结构如图 12.3 所示。



名称	修改日期	类型
ant	2012-07-24 17:12	文件夹
checkstyle	2012-07-24 17:12	文件夹
docfiles	2012-07-24 17:12	文件夹
experimental	2012-07-24 17:12	文件夹
lib	2012-07-25 8:55	文件夹
source	2012-07-24 17:13	文件夹
swt	2012-07-24 17:13	文件夹
tests	2012-07-24 17:13	文件夹
ChangeLog	2011-11-20 22:40	文件
jfreechart-1.0.14-demo.jar	2011-11-20 22:40	JAR 文件
licence-LGPL.txt	2011-11-20 22:40	文本文档
NEWS	2011-11-20 22:40	文件
OnBridge11small.png	2011-11-20 22:01	PNG 文件
pom.xml	2011-11-20 22:40	XML 文档
README.txt	2011-11-20 22:40	文本文档

图 12.3 jfreechart-1.0.14 文件结构

其中，jfreechart-1.0.14-demo.jar 文件为 JFreeChart 组件提供的演示文件，运行此文件将可以看到 JFreeChart 组件制作的各种图表的样式及效果；source 文件夹为 JFreeChart 的源代码文件夹，在此文件夹中可以查看到 JFreeChart 组件的源代码；lib 文件夹为 JFreeChart 的支持类库，在本章内容中，主要用到 jfreechart-1.0.14.jar 和 jcommon-1.0.17.jar 两个 jar 包。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

12.2　JFreeChart的核心对象

JFreeChart 组件对绘制图表的细节进行了封装，它对外提供了绘制图形及设置图形属性的接口与方法。所以，在 JFreeChart 组件的使用之中，不必关心它所绘制的图形报表是如何实现的，但要了解 JFreeChart 组件的结构并掌握其核心对象。

12.2.1　制图对象

JFreeChart 类是一个制图对象，它代表着一种制图类型。例如，创建一个柱形图，首先需要创建一个柱形图的制图对象 JFreeChart；创建一个饼形图，需要创建个饼形图的制图对象 JFreeChart。在制图过程中，只有在创建制图对象 JFreeChart 后，才可以生成实际的图片。

在 JFreeChart 类中，可以设置所生成图片的边界、字体、背景、透明度等属性，其常用方法及说明如表 12.1 所示。

表 12.1　JFreeChart 类常用方法及说明

方　法	说　明
public void setAntiAlias(boolean flag)	设置字体模糊边界
public void setBackgroundImage(Image image)	设置背景图片
public void setBackgroundImageAlignment(int alignment)	设置背景图片对齐方式，其参数常量定义在 org.jfree.ui.Align 类中
public void setBackgroundImageAlpha(float alpha)	设置背景图片透明度
public void setBackgroundPaint(Paint paint)	设置背景颜色
public void setBorderPaint(Paint paint)	设置边界线条颜色
public void setBorderVisible(boolean visible)	设置边界线条是否可见

12.2.2　制图工厂对象

在生成图形报表时，制图对象 JFreeChart 是必不可少的，它可以直接通过 new 关键字进行实例化，也可以通过制图工厂 ChartFacroty 类进行实例化。当使用 new 关键字进行实例化时，需要设置大量的属性信息，因为 JFreeChart 组件提供图表的种类很多，对于每一种图表都要进行特殊的设置，非常烦琐。所以在使用过程中，一般都使用制图工厂 ChartFacroty 类进行创建。

制图工厂 ChartFacroty 是一个抽象类，它不能被实例化，但此类之中提供创建各种制图对象的方法，如创建柱形图对象、区域图对象、饼形图对象、折线图对象等方法，这些方法都是静态的方法，可直接创建 JFreeChart 对象，并且是属于某一种具体的图表类型的 JFreeChart 对象，其使用非常方便，它的常用方法及说明如表 12.2 所示。

表 12.2　ChartFacroty 类常用方法及说明

图表类型	方　法	说　明
柱形图	public static JFreeChart createBarChart()	创建一个常规的柱形图对象
	public static JFreeChart createBarChart3D()	创建一个 3D 效果的柱形图对象
饼形图	public static JFreeChart createPieChart()	创建一个常规的饼形图对象
	public static JFreeChart createPieChart3D()	创建一个 3D 效果的饼形图对象

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（续）

图表类型	方　法	说　明
区域图	public static JFreeChart createAreaChart()	创建一个常规的区域图对象
折线图	public static JFreeChart createLineChart()	创建一个常规的折线图对象
	public static JFreeChart createLineChart3D()	创建一个 3D 效果的折线图对象
时序图	public static JFreeChart createTimeSeriesChart()	创建一个常规的时序图对象

除表 12.2 中所列的方法外，ChartFacroty 类还有很多创建各种类型制图对象的方法，在此不一一进行罗列。对于同种类型的制图对象，ChartFacroty 类为其提供一个或多个方法，表 12.2 所示为常规制图对象与 3D 效果的制图对象的方法。可见，制图工厂 ChartFacroty 类为创建制图对象提供了多个灵活的方法。

ChartFacroty 类可以理解为是一个生产制图对象 JFreeChart 的工厂，当需要用到某一种类型的制图对象时，通过此工厂进行获取。

例如，下面的代码将通过 ChartFactory 类的 createPieChart3D() 方法创建一个 3D 效果的饼形图 JFreeChart 对象。

```
JFreeChart chart = ChartFactory.createPieChart3D(
    "饼形图",                    // 图表的标题
    initPieData(),               // 饼形图的数据集对象
    true,                        // 是否显示图例
    true,                        // 是否显示提示文本
    false);                      // 是否生成超链接
```

12.2.3　数据集合对象

在 JFreeChart 组件的图形报表技术应用中，绘制一个图表需要一定的数据，JFreeChart 组件通过提供的数据进行计算并绘制出图表信息。由于在数据的分析计算中并不是单一的数值，所以绘制图表要为 JFreeChart 组件提供数据集合。

数据集合对象是用于装载绘制图表所需要数据的集合。JFreeChart 组件针对不同图表类型提供了不同的数据集合对象，它们所具有的作用也是不同的。本节内容将以常用的数据集合对象为例，来讲解 JFreeChart 中的数据集合对象。在学习 JFreeChart 数据集合对象前，先来了解一下常用数据集合对象之间的关系，其类图如图 12.4 所示。

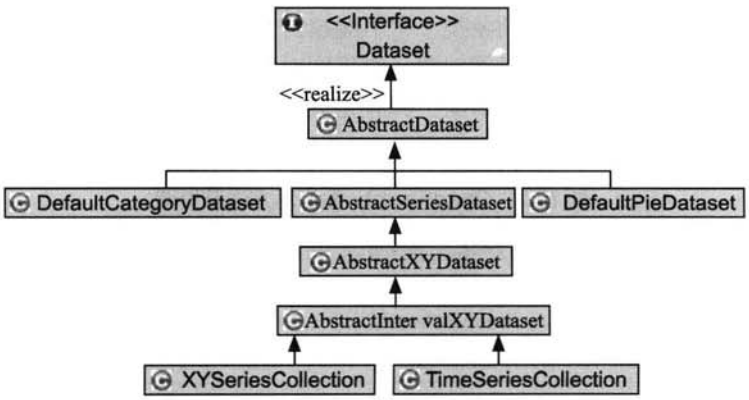


图 12.4　常用数据集合间的关系

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Dataset 接口是数据集合的核心对象，从图 12.4 可以看出，所有数据集合对象都直接或间接地实现了此接口。图中类名以 Abstract 开头的类均为抽象类，它们并不能实例化，但为其子类提供公共属性与方法，其中 DefaultCategoryDataset 类、DefaultPieDataset 类、XYSeriesCollection 类与 TimeSeriesCollection 类是经常用到的数据集合对象，其说明如表 12.3 所示。

表 12.3　常用数据集合对象及说明

数据集合对象	说　明
DefaultCategoryDataset 类	默认的类别数据集合对象，可用于创建柱形图、区域图数据集合等
DefaultPieDataset 类	默认的饼形图数据集合对象，可用于创建饼形图数据集合
XYSeriesCollection 类	描述坐标轴序列类型的数据集合，可用于创建折线图等数据集合
TimeSeriesCollection 类	描述时间序列的数据集合，可用于创建时序图等数据集合

数据集合是数据集的封装对象，在 JFreeChart 组件的使用中，绘制每一种图形都需要用到数据集合对象。图 12.4 中只介绍了常用的数据集合关系，除了这些数据集合外，JFreeChart 还针对不同图表提供了各类数据集合，由于篇幅原因，这里不一一介绍，其使用方法请参阅 JFreeChart 组件的 API 文档。

例如，下面的代码将创建一个 DefaultCategoryDataset 类型的数据集合，并调用 addValue() 方法向数据集合中添加一个数据。

```
// 创建数据集合
DefaultCategoryDataset dataSet = new DefaultCategoryDataset();
// 向数据集合中添加数据
dataSet.addValue(100, "长春", "土豆");
```

12.2.4　绘图区对象

数据集合所生成的数据图表可以经由绘图区对象进行属性设置，例如背景色、透明度等样式。绘图区对象是 JFreeChart 组件中的一个重要对象，它由 Plot 类定义，在使用过程中可以通过此类设置绘图区属性及样式，其常用方法及说明如表 12.4 所示。

表 12.4　Plot 类常用方法及说明

方　法	说　明
public void setBackgroundImage(Image image)	设置数据区的背景图片
public void setBackgroundImageAlignment(int alignment)	设置数据区的背景图片对齐方式（参数常量在 org.jfree.ui.Align 类中定义）
public void setBackgroundAlpha(float alpha)	设置数据区的背景透明度，范围为 0.0 ~ 1.0
public void setForegroundAlpha(float alpha)	设置数据区的前景透明度，范围为 0.0 ~ 1.0
public void setDataAreaRatio(double ratio)	设置数据区占整个图表区的百分比
public void setOutLinePaint(Paint paint)	设置数据区的边界线条颜色
public void setNoDataMessage(String message)	设置没有数据时显示的消息

JFreeChart 所能生成的图形报表是多种多样的，仅仅一个 Plot 类并不能满足绘图区样式的

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

设置，在对不同类型图形进行设置时，可以通过 Plot 的子类进行实现，其常用子类的类图如图 12.5 所示。

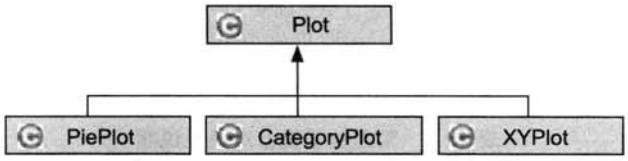


图 12.5　Plot 常用子类类图

1．PiePlot 类

PiePlot 类是 Plot 类的子类，主要用于描述 PieDataset 数据集合类型的图表，通常使用此类来绘制一个饼形图，其常用方法及说明如表 12.5 所示。

表 12.5　PiePlot 类常用方法及说明

方　法	说　明
public void setDataset(PieDataset dataset)	设置绘制图表所需要的数据集合
public void setCircular(boolean flag)	设置饼形图是否一定是正圆
public void setStartAngle(double angle)	设置饼形图的初始角度
public void setDirection(Rotation direction)	设置饼形图的旋转方向
public void setExplodePercent(int section,double percent)	设置抽取图表的部分及距离
public void setLabelFont(Font font)	设置分类标签字体（3D 效果下无效）
public void setLabelPaint(Paint paint)	设置分类标签字体颜色（3D 效果下无效）

2．CategoryPlot 类

CategoryPlot 类主要用于描述 CategoryDataset 数据集合类型的图表，是 Plot 类的子类，它支持折线图、区域图等，其常用方法及说明如表 12.6 所示。

表 12.6　CategoryPlot 类常用方法及说明

方　法	说　明
public void setDataset(CategoryDataset dataset)	设置绘制图表所需要的数据集合
public void setColumnRenderingOrder(SortOrder order)	设置数据分类的排序方式
public void setAxisOffset(Spacer offset)	设置坐标轴到数据区的间距
public void setOrientation(PlotOrientation orientation)	设置数据区的方向（横向或纵向）
public void setDomainAxis(CategoryAxis axis)	设置数据区的分类轴
public void setRangeAxis(ValueAxis axis)	设置数据区的数据轴
public void addAnnotation(CategoryAnnotation annotation)	设置数据区的注释

3．XYPlot 类

XYPlot 类是 Plot 类的子类，主要用于描述 XYDataset 数据集合类型的图表，此类可以有 0 个或多个数据集合，并且每一个数据集合可以与一个渲染对象相关联，其常用方法及说明如表 12.7 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 12.7　XYPlot 类常用方法及说明

方　法	说　明
public ValueAxis getDomainAxis()	返回 X 轴
public ValueAxis getRangeAxis()	返回 Y 轴
public void setDomainAxis(ValueAxis axis)	设置 X 轴
public void setRangeAxis(ValueAxis axis)	设置 Y 轴

12.2.5　坐标轴对象

在 JFreeChart 组件中，涉及坐标轴类型的图表，其样式与属性由坐标轴对象 Axis 类进行控制，此类是坐标轴对象的父类，其常用方法及说明如表 12.8 所示。

表 12.8　Axis 类常用方法及说明

方　法	说　明
public void setVisible(boolean flag)	设置坐标轴是否可见
public void setAxisLinePaint(Paint paint)	设置坐标轴线条颜色，此设置在 3D 效果下无效
public void setAxisLineVisible(boolean visible)	设置坐标轴线条是否可见
public void setLabel(String label)	设置坐标轴标题
public void setLabelFont(Font font)	设置坐标轴标题字体
public void setLabelPaint(Paint paint)	设置坐标轴标题颜色
public void setLabelAngle(double angle)	设置坐标轴标题旋转角度

JFreeChart 组件针对不同类型的图表对象，提供了不同类型的坐标轴对象，由 Axis 类的子类进行扩展，其常用子类的类图如图 12.6 所示。

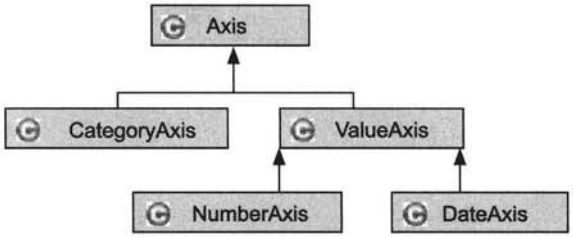


图 12.6　常用坐标轴对象及关系

1．CategoryAxis 类

CategoryAxis 类是 Axis 类的子类，主要用于对分类轴的相关属性进行设置，其常用方法及说明如表 12.9 所示。

表 12.9　CategoryAxis 类常用方法及说明

方　法	说　明
public void setCategoryMargin(double margin)	设置分类轴边距
public void setLowerMargin(double margin)	设置分类轴下边距或左边距
public void setUpperMargin(double margin)	设置分类轴上边距或右边距
public void setVerticalCategoryLabels(boolean flag)	设置分类轴标题是否旋转到垂直
public void setMaxCategoryLabelWidthRatio(float ratio)	设置分类轴分类标签的最大宽度

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2. ValueAxis 类

ValueAxis 类是 Axis 类的子类，主要用于对数据轴的相关属性进行设置，它还是 NumberAxis 类与 DateAxis 类的父类，其常用方法及说明如表 12.10 所示。

表 12.10 ValueAxis 类常用方法及说明

方 法	说 明
public void setAutoRange(boolean auto)	设置数据轴数据范围是否为自动
public void setFixedAutoRange(double length)	设置数据轴固定数据范围
public void setInverted(boolean flag)	设置数据轴是否反向
public void setLowerMargin(double margin)	设置数据轴下边距或左边距
public void setUpperMargin(double margin)	设置数据轴上边距或右边距
public void setLowerBound(double min)	设置数据轴上的显示最小值
public void setUpperBound(double max)	设置数据轴上的显示最大值

3. NumberAxis 类

NumberAxis 类主要用于对数值类型数据轴的相关属性进行设置，是 ValueAxis 类的子类，其常用方法及说明如表 12.11 所示。

表 12.11 NumberAxis 类常用方法及说明

方 法	说 明
public void setAutoRangeIncludesZero(boolean flag)	设置是否强制在自动选择的数据范围中包含 0
public void setAutoRangeStickyZero(boolean flag)	设置是否强制在整个数据轴中包含 0，即使 0 不在数据范围中
public void setNumberFormatOverride(NumberFormat formatter)	设置数据轴数据标签的显示格式

4. DateAxis 类

DateAxis 类主要用于对日期轴的相关属性进行设置，是 ValueAxis 类的子类，其常用方法及说明如表 12.12 所示。

表 12.12 DateAxis 类常用方法及说明

方 法	说 明
public void setMaximumDate(Date maximumDate)	设置日期轴上的最小日期
Public void setMinimumDate(Date minimumDate)	设置日期轴上的最大日期
public void setDateFormatOverride(DateFormat formatter)	设置日期轴上日期标签的显示格式
public void setTickUnit(DateTickUnit unit)	设置日期轴的日期标签

12.2.6 图片渲染对象

图片渲染对象用于渲染和显示图表，它在图表的显示效果方面起着很大的作用。在 JFreeChart 组件中，渲染对象定义为 AbstractRenderer 类，此类是所有渲染对象的父类，但它是一个抽象类，其常用方法及说明如表 12.13 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 12.13　AbstractRenderer 类常用方法及说明

方　法	说　明
public void setItemLabelAnchorOffset(double offset)	设置数据标签与数据点的偏移
public void setItemLabelsVisible(boolean visible)	设置数据标签是否可见
public void setItemLabelFont(Font font)	设置数据标签的字体
public void setItemLabelPaint(Paint paint)	设置数据标签的字体颜色
public void setOutLinePaint(Paint paint)	设置图形边框的线条颜色
public void setPaint(Paint paint)	设置所有分类图形的颜色
public void setSeriesItemLabelsVisible(int series,boolean visible)	设置分类的数据标签是否可见
public void setSeriesItemLabelFont(int series,Font font)	设置分类的数据标签的字体
public void setSeriesItemLabelPaint(int series,Paint paint)	设置分类的数据标签的字体颜色

在图片渲染对象中，JFreeChart 组件同样对 AbstractRenderer 类进行了扩展，在使用过程中可根据实际需要，选择合适的 AbstractRenderer 类的子类对象。

12.3　JFreeChart的应用

JFreeChart 组件是 Java 领域中一个强大的图形报表组件，在 Web 项目中经常使用此组件，其制图过程主要分为三步来实现：配置 JFreeChart、创建数据集合与 JFreeChart 实例、设置图表的相关属性。

12.3.1　如何获取图片

JFreeChart 组件能够生成 .jpeg、.png 格式的图片，其输出方式可以直接存储在硬盘之中，也可以交给 JFreeChart 组件进行管理。在 Web 项目的应用中，所生成的图形报表一般均为动态图表，如果对每次生成的图表都进行直接存储，将会产生大量的垃圾文件，需要做一定清理工作。因此，由 JFreeChart 组件管理所生成的图片是一个不错的选择，它不仅可以提供图片的路径，而且在查看图片后，JFreeChart 组件会自动进行清理工作。

Step 1　配置 JFreeChart。

JFreeChart 组件提供了一个 Servlet 文件用于获取生成的图片，此 Servlet 文件存在于 JFreeChart 组件包中，所以在使用过程中，需要将其配置到 Web.xml 文件中，其配置方法如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
   <!-- 注册 Servlet -->
   <servlet>
          <!-- Servlet 描述信息 -->
          <description>JFreeChart</description>
          <!-- Servlet 的名称 -->
          <servlet-name>DisplayChart</servlet-name>
          <!-- Servlet 类的完整类名 -->
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            <servlet-class>org.jfree.chart.servlet.DisplayChart</servlet-class>
    </servlet>
    <!-- Servlet 映射 -->
    <servlet-mapping>
            <!-- Servlet 名称 -->
            <servlet-name>DisplayChart</servlet-name>
            <!-- 访问 URL 地址 -->
            <url-pattern>//DisplayChart</url-pattern>
    </servlet-mapping>
</web-app>
```

从上述代码可以看出，它的配置与普通的 Servlet 配置是完全相同的。

Step 2) 获取图片。

在输出图片之前，首先要生成 JFreeChart 组件所绘制的图片，此操作通过调用 ServletUtilities 类的 saveChartAsJPEG() 方法进行生成，它返回一个 .jpeg 格式的图片。

saveChartAsJPEG() 方法的语法格式如下：

```
public static String saveChartAsJPEG(JFreeChart chart, int width,
                                      int height, HttpSession session)
            throws IOException
```

- chart：制图对象 JFreeChart。
- width：所生成图片的宽度。
- height：所生成图片的高度。
- session：HttpSession 对象。

通过此方法生成图片后，调用已注册的 JFreeChart 提供的 Servlet 类 DisplayChart，即可获取图片的相对路径。

例如，在 JSP 页面中获取生成的图片可以使用下面的代码。

```
<%
    String fileName = ServletUtilities.saveChartAsJPEG(ChartUtil.
    createChart(), 450, 300, session);
String graphURL = request.getContextPath() + "/DisplayChart?filename=" + fileName;
%>
```

上述代码生成了一个宽为 450 像素、高为 300 像素的图片，其文件名 fileName，它的路径为 graphURL，通过此路径即可对所生成的图片进行访问。

12.3.2 创建数据集合与 JFreeChart 实例

JFreeChart 类是一个制图对象，创建时需要一个数据集合对象，在拥有数据集合后即可通过制图工厂进行创建。JFreeChart 提供的数据集合是多种多样的，在实际应用中要选择合适的数据集合对象，对于不同的数据集合对象，其创建及添加数据的方法也是不同的。

【例 12.1】 使用柱形图统计 Java 图书销量，通过 DefaultCategoryDataset 数据集合对象创建柱形图制图对象，其关键代码如下：

实例位置：光盘 \MR\ 源码 \ 第 12 章 \12.1

```
public class ChartUtil {
    /**
     * 创建数据集合
     * @return CategoryDataset 对象
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
     */
    public static CategoryDataset createDataSet() {
            DefaultCategoryDataset dataSet = new DefaultCategoryDataset();
                                            // 实例化 DefaultCategoryDataset 对象
            // 向数据集合中添加数据
            dataSet.addValue(550, "JAVA 图书", "Java SE 类");
            dataSet.addValue(100, "JAVA 图书", "Java ME 类");
            dataSet.addValue(960, "JAVA 图书", "Java EE 类");
            return dataSet;
    }
    /**
     * 创建 JFreeChart 对象
     * @return JFreeChart 对象
     */
    public static JFreeChart createChart() {
            StandardChartTheme standardChartTheme = new StandardChartTheme("CN");
                                                                    // 创建主题样式
            standardChartTheme.setExtraLargeFont(new Font("隶书", Font.BOLD, 20));
                                                                    // 设置标题字体
            standardChartTheme.setRegularFont(new Font("微软雅黑", Font.PLAIN, 15));
                                                                    // 设置图例的字体
            standardChartTheme.setLargeFont(new Font("微软雅黑", Font.PLAIN, 15));
                                                                    // 设置轴向的字体
            ChartFactory.setChartTheme(standardChartTheme);
                                                                    // 设置主题样式
            // 通过 ChartFactory 创建 JFreeChart
            JFreeChart chart = ChartFactory.createBarChart3D(
                            "JAVA 图书销量统计",              // 图表标题
                            "JAVA 图书",                      // 横轴标题
                            "销量（本）",                     // 纵轴标题
                            createDataSet(),                  // 数据集合
                            PlotOrientation.VERTICAL,         // 图表方向
                            false,                            // 是否显示图例标识
                            false,                            // 是否显示 tooltips
                            false);                           // 是否支持超链接
            return chart;
    }
}
```

ChartUtil 类是一个自定义的制图工具类，其中 createDataSet() 方法用于创建柱形图所需要的数据集合，它返回 CategoryDataset 对象。CreateChart() 用于创建制图对象 JFreeChart，在此方法中通过 ChartFactory 对象的 createBarChart3D() 方法创建一个 3D 效果的柱形图对象，并将其返回。

在 JSP 页面中通过 JFreeChart 组件提供的 Servlet 类 DisplayChart 进行图片获取，实例在 index.jsp 页面中输出图片，其关键代码如下：

```
<%@ page language="java" contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="org.jfree.chart.servlet.ServletUtilities,com.wgh.util.
ChartUtil"%>
<!DOCTYPE HTML>
<html>
  <head>
    <title>Java 图书销量统计 </title>
  </head>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    <body>
      <%
            String fileName = ServletUtilities.saveChartAsJPEG(ChartUtil.
            createChart(),450,300,session);
            String graphURL = request.getContextPath() + "/DisplayChart?filename="
            + fileName;
      %>
      <img src="<%=graphURL%>" border="1">
    </body>
</html>
```

在输出图片之前，首先要生成JFreeChart组件所绘制的图片，此操作通过调用ServletUtilities类的saveChartAsJPEG()方法进行生成，它返回一个JPEG格式的图片名称。

在获取此图片名称后，通过代码“request.getContextPath() + "/DisplayChart?filename=" + fileName”即可获取图片的路径，其运行结果如图12.7所示。

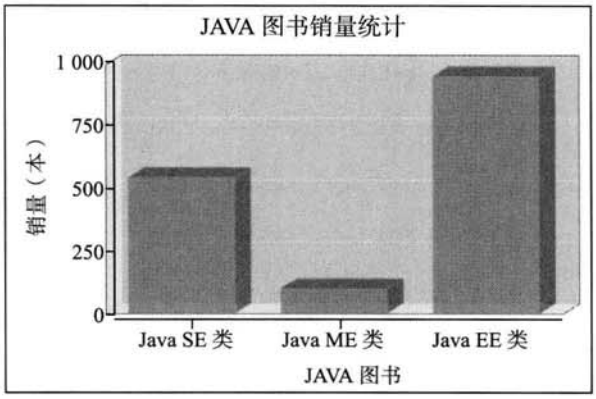


图12.7 柱形图统计Java图书销量

12.3.3 图表相关属性的设置

通常情况下，为了使所生成的图片美观、大方，需要对所生成的图片进行一定的设置，如制图对象设置、绘图区设置、坐标轴设置及图片渲染等。

【例12.2】 利用柱形图统计Java类图书第一季度至第四季度销量，创建数据集合对象DefaultCategoryDataset，并添加四个季度的销量数据，其关键代码如下：

实例位置：光盘\MR\源码\第12章\12.2

```
    public static CategoryDataset createDataSet() {
            // 实例化 DefaultCategoryDataset 对象
            DefaultCategoryDataset dataSet = new DefaultCategoryDataset();
            // 添加第一季度数据
            dataSet.addValue(6000, "第一季度", "Java SE 类");
            dataSet.addValue(3000, "第一季度", "Java ME 类");
            dataSet.addValue(12000, "第一季度", "Java EE 类");
            // 添加第二季度数据
            dataSet.addValue(8000, "第二季度", "Java SE 类");
            dataSet.addValue(4000, "第二季度", "Java ME 类");
            dataSet.addValue(6000, "第二季度", "Java EE 类");
            // 添加第三季度数据
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        dataSet.addValue(5000, "第三季度", "Java SE 类");
        dataSet.addValue(4000, "第三季度", "Java ME 类");
        dataSet.addValue(8000, "第三季度", "Java EE 类");
        //添加第四季度数据
        dataSet.addValue(8000, "第四季度", "Java SE 类");
        dataSet.addValue(2000, "第四季度", "Java ME 类");
        dataSet.addValue(9000, "第四季度", "Java EE 类");
        return dataSet;
    }
```

createDataSet() 方法定义在 ChartUtil 类中，它是一个自定义的制图工具类。在此类中，还包含创建制图对象的方法 createChart()，用于创建柱形图对象，其关键代码如下：

```
public static JFreeChart createChart() {
    //通过 ChartFactory 创建 JFreeChart
    JFreeChart chart = ChartFactory.createBarChart3D(
                    "JAVA 图书销量统计",              //图表标题
                    "JAVA 图书",                     //横轴标题
                    "销量（本）",                    //纵轴标题
                    createDataSet(),                 //数据集合
                    PlotOrientation.VERTICAL,        //图表方向
                    true,                            //是否显示图例标识
                    false,                           //是否显示 tooltips
                    false);                          //是否支持超链接
    Image image = null;                              //背景图片
    try {
            image = ImageIO.read(ChartUtil.class.getResource("test.JPG"));
                                                     //创建背景图片
    } catch (IOException e) {
            e.printStackTrace();
    }
    chart.getTitle().setFont(new Font("隶书",Font.BOLD,25));
                                                     //设置标题字体
    chart.getLegend().setItemFont(new Font("宋体",Font.PLAIN,12));
                                                     //设置图例类别字体
    chart.setBorderVisible(true);
                                                     //设置显示边框
    //实例化 TextTitle 对象
    TextTitle subTitle = new TextTitle("2012 年 Java 类图书全国销量统计（JavaSE、JavaME、
    JavaEE）");
    subTitle.setVerticalAlignment(VerticalAlignment.BOTTOM);
                                                     //设置居中显示
    chart.addSubtitle(subTitle);                     //添加子标题
    CategoryPlot plot = chart.getCategoryPlot();     //获取绘图区对象
    plot.setForegroundAlpha(0.8F);                   //设置绘图区前景色透明度
    plot.setBackgroundAlpha(0.5F);                   //设置绘图区背景色透明度
    plot.setBackgroundImage(image);                  //设置绘图区背景图片
    CategoryAxis categoryAxis = plot.getDomainAxis();//获取坐标轴对象
    categoryAxis.setLabelFont(new Font("微软雅黑",Font.PLAIN,12));
                                                     //设置坐标轴标题字体
    categoryAxis.setTickLabelFont(new Font("微软雅黑",Font.PLAIN,12));
                                                     //设置坐标轴标尺值字体
    categoryAxis.setCategoryLabelPositions(CategoryLabelPositions.UP_45);
                                                     //设置坐标轴标题旋转角度
    ValueAxis valueAxis = plot.getRangeAxis();       //获取数据轴对象
    valueAxis.setLabelFont(new Font("微软雅黑",Font.PLAIN,12));
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                                                        //设置数据轴字体
    BarRenderer3D renderer = new BarRenderer3D();       //获取图片渲染对象
    renderer.setItemMargin(0.32);                       //设置柱子间的间距
    plot.setRenderer(renderer);                         //设置图片渲染对象
    return chart;
}
```

在此方法中，首先创建了一个 3D 效果的柱形图对象，在创建之后分别对制图对象、绘图区对象、坐标轴对象、图片渲染对象等属性进行设置，并添加了一个子标题。实例中通过 index.jsp 页面进行输出，其运行结果如图 12.8 所示。

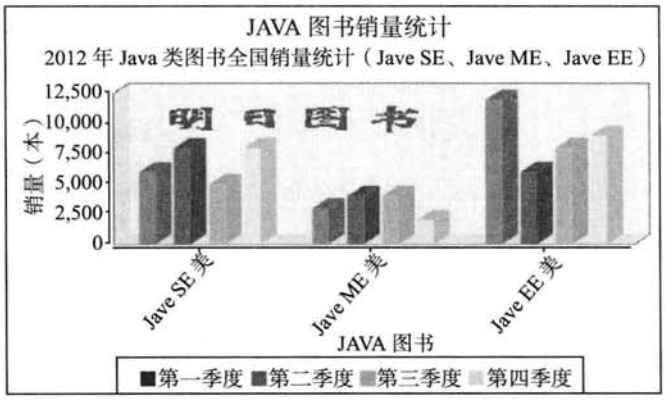


图 12.8　利用柱形图统计 Java 图书销量

12.3.4　JFreeChart 内置 JDBC 的使用

在实际开发过程中，数据集合中的数据大部分来自数据库，所以在制图前，需要进行数据库操作来获取数据。JFreeChart 组件对这一过程进行了封装，通过自定义的 SQL 语句就可获取已封装好的数据集合对象。

常用的 JDBC 数据集合对象有三种，分别为 JDBCCategoryDataSet（JDBC 填充类别数据集合）、JDBCPieDataSet（JDBC 填充饼形图数据集合）、JDBCXYDataset（JDBC 填充坐标轴数据集合）。它们的使用方法都非常简单，例如，创建 JDBCPieDataSet 对象的方法如下：

```
public JDBCPieDataset(String url,
                      String driverName,
                      String user,
                      String password)
    throws SQLException, ClassNotFoundException
```

- url：数据库连接的 url。
- driverName：数据库的驱动类。
- user：数据库连接用户名。
- password：数据库连接密码。

创建了 JDBCPieDataset 对象后，可以通过 executeQuery() 方法查询数据库，其入口参数为 String 类型的 SQL 语句，执行此方法后，将返回拥有数据的数据集合对象。

【例 12.3】 使用饼形图对商城各类别商品的月销量进行统计。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

实例位置：光盘\MR\源码\第12章\12.3

在本实例中，数据集合由 JFreeChart 组件查询数据库后自动生成，其数据表 tb_shop 中数据如图 12.9 所示。

id	category	val
1	数码产品	300
2	家用电器	110
3	日常用品	160
4	服装	240
5	水果蔬菜	220
6	其他	50

图 12.9　表 tb_shop 的数据

数据集合与制图对象的创建定义在 ChartUtil 类中，它是一个自定义制图工具类，用于创建数据集合及制图对象，其关键代码如下：

```
public class ChartUtil {
    /**
     * 查询数据库并初始化数据集合
     * @return PieDataset 对象
     */
    public static PieDataset initPieData() {
            String driverName = "com.mysql.jdbc.Driver"; // 数据库驱动
            String url = "jdbc:mysql://localhost:3306/db_database12";
                                                        // 数据库连接 url
            String user = "root";                       // 数据库用户名
            String password = "111";                    // 数据库密码
            JDBCPieDataset dataset = null;              //数据集合
            try {
                    dataset = new JDBCPieDataset(url, driverName, user, password);
                    // 通过 JDBC 创建数据集合
                                String query = "select category,val from tb_
                                shop";                  // SQL 语句
                    dataset.executeQuery(query);
                    // 查询并向数据集合中添加数据
                    dataset.close();
                    // 关闭数据库连接
            } catch (Exception e) {
                    e.printStackTrace();
            }
            return dataset;
    }
    /**
     * 创建饼形图实例
     * @return JFreeChart 对象
     */
    public static JFreeChart createChart() {
            // 创建 3D 饼型图表
            JFreeChart chart = ChartFactory.createPieChart3D(
                            "XX 商城月销量统计 ",      // 图表的标题
                            initPieData(),          // 饼形图的数据集对象
                            true,                   // 是否显示图例
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                                true,                    // 是否显示提示文本
                                false);                  // 是否生成超链接
            chart.getTitle().setFont(new Font("隶书",Font.BOLD,25));
                                                         // 设置标题字体
            chart.getLegend().setItemFont(new Font("宋体",Font.BOLD,15));
                                                         // 设置图例类别字体
            PiePlot plot = (PiePlot) chart.getPlot();
                                                         // 获得绘图区对象
            plot.setForegroundAlpha(0.5f);               // 设置前景透明度
            plot.setLabelFont(new Font("宋体",Font.PLAIN,12));
                                                         // 设置分类标签的字体
            plot.setCircular(true);                      // 设置饼形为正圆
            // 设置分类标签的格式
            plot.setLabelGenerator(new StandardPieSectionLabelGenerator("{0}={2}",
                                NumberFormat.getNumberInstance(),
                                NumberFormat.getPercentInstance()));
            return chart;
        }
    }
```

initPieData() 方法用于创建数据集合对象，在此方法中，使用 JDBCPieDataset 类通过 JDBC 查询数据库获取数据集合对象。

createChart() 方法用于创建饼形图制图对象，在创建制图对象后对其相关属性进行了设置。

注意 PiePlot 类的 setLabelGenerator() 方法用于设置分类标签的格式，其参数为 StandardPieSectionLabelGenerator 对象，此对象的入口参数 "{0}={2}" 用于指定类别名称及所占有的百分比，{0} 代表类别名称，{2} 代表百分比。

编写了 ChartUtil 类后，通过 index.jsp 页面查看图片，其关键代码如下：

```
<%@ page language="java" contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="org.jfree.chart.servlet.ServletUtilities,com.wgh.util.
ChartUtil"%>
<!DOCTYPE HTML>
<html>
  <head>
    <title>XX 商城月销量统计 </title>
  </head>
  <body>
    <%
    String fileName = ServletUtilities.saveChartAsJPEG(ChartUtil.
    createChart(),500,300,session);
    String graphURL = request.getContextPath() + "/DisplayChart?filename="
    + fileName;
    %>
    <div align="center">
    <img src="<%=graphURL%>" border="1">
    </div>
  </body>
</html>
```

运行本实例，将显示如图 12.10 所示的运行结果。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

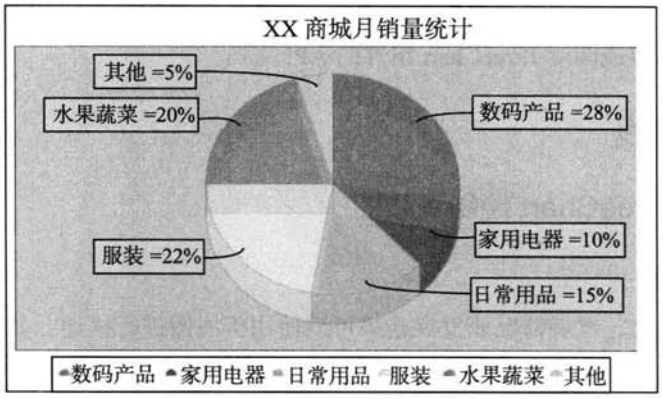


图 12.10　饼形图分析商城销量

12.3.5　中文乱码的解决方案

由于 JFreeChart 组件的版本、操作平台、JDK 的设置等各不相同，在使用 JFreeChart 组件时，可能出现中文乱码的现象。遇到此问题时，可通过设置乱码文字的字体进行解决，在此笔者提供两种解决此问题的方法。

（1）设置主题样式

在制图前，创建主题样式并指定样式中的字体，通过 ChartFactory 的 setChartTheme() 方法设置主题样式，在指定制图样式后，ChartFactory 对象创建的制图对象将按此样式进行显示，图表中的文字将按指定的字体进行显示。

例如：

```
StandardChartTheme standardChartTheme = new StandardChartTheme("CN");
                                                    //创建主题样式
standardChartTheme.setExtraLargeFont(new Font("隶书", Font.BOLD, 20));
                                                    //设置标题字体
standardChartTheme.setRegularFont(new Font("宋体", Font.PLAIN, 15));
                                                    //设置图例的字体
standardChartTheme.setLargeFont(new Font("宋体", Font.PLAIN, 15));
                                                    //设置轴向的字体
ChartFactory.setChartTheme(standardChartTheme); //应用主题样式
```

通过上述代码设置主题样式后，再通过 ChartFactory 创建 JFreeChart 的对象，可以解决中文乱码问题。

（2）指定乱码文字的字体

此方法通过指定制图对象中的中文字体解决中文乱码问题，在图中任何用到中文的地方，都要对字体进行设置，此操作将涉及 JFreeChart 对象、Plot 对象、坐标轴对象的属性设置。

例如：

```
JFreeChart chart = null;
… // 省略部分代码
chart.getTitle().setFont(new Font("隶书",Font.BOLD,25));          // 设置标题字体
chart.getLegend().setItemFont(new Font("宋体",Font.BOLD,15));     // 设置图例类别字体
PiePlot plot = (PiePlot) chart.getPlot();                         // 获得绘图区对象
plot.setLabelFont(new Font("宋体",Font.PLAIN,12));                // 设置分类标签的字体
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

不同的制图对象类型，其相关字体的设置可能存在差异，对于各对象中所用到字体设置，请参阅例 12.2 的内容或阅读 JFreeChart 组件的 API 文档。

12.4 上机实践

12.4.1 利用 JFreeChart 绘制折线图

➤➤➤ 题目描述

对于数据的变化、发展情况或发展趋势可以使用折线图进行描述，其显示方式非常直观。本上机实践尝试使用折线图对 2008 年至 2012 年智能手机软件下载数量进行统计，其显示效果分为普通样式及 3D 样式。运行结果如图 12.11 和图 12.12 所示。

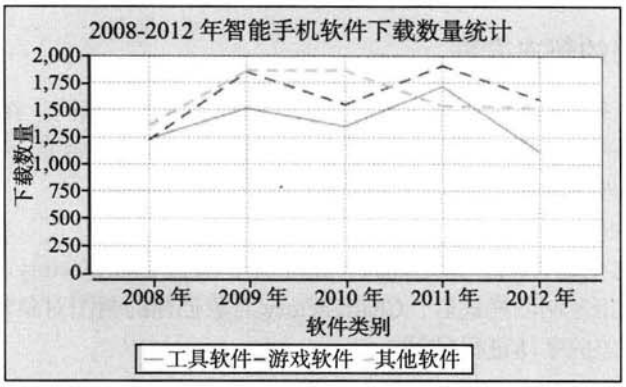


图 12.11 普通样式

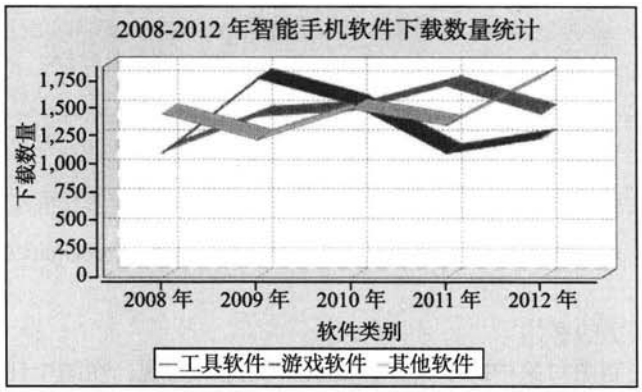


图 12.12 3D 样式

说明 本实例要求数据集合的数据随机生成，每次显示结果并非一致。

➤➤➤ 技术指导

很多时候在一个图表中包含的图例数目是不同的，这就要求动态生成图例名称数组。本例

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

的实现思路是首先获得要统计的商品数目，然后初始化图例数组，在封装统计数据的同时封装图例名称。

JFreeChart 组件提供了制图工厂类 ChartFactory，该类的静态方法 createLineChart() 用于创建折线图的制图对象，其语法格式如下：

```
public static JFreeChart createLineChart(String title,
        String categoryAxisLabel,
        String valueAxisLabel,
        CategoryDataset dataset,
        PlotOrientation orientation,
        boolean legend,
        boolean tooltips,
        boolean urls)
```

createLineChart() 方法的参数说明如表 12.14 所示。

表 12.14　createLineChart() 方法的参数说明

参　数	说　明	参　数	说　明
title	图表的标题	categoryAxisLabel	横轴上的标签文字
valueAxisLabel	数据值轴上的标签文字	dataset	制图所需的数据集合
orientation	图表方向	legend	是否显示图例
tooltips	是否显示说明文字	urls	是否生成超链接

说明　使用 createLineChart() 方法创建的制图对象用于绘制的是普通的折线图，要绘制 3D 的折线图，需要使用 createLineChart3D() 方法，该方法与 createLineChart() 方法除了方法名不同以外，其他参数均相同。

12.4.2　利用 JFreeChart 绘制区域图

➤➤➤ 题目描述

区域图可以用于描述数据对比情况、部分与整体的关系、数据的变化程度等，其表现形式多样。本上机实践尝试通过统计某门户网站流量，使用区域图对新闻网站、娱乐网站、体育网站三个类别进行对比分析，运行结果如图 12.13 所示。

说明　本实例要求数据集合的数据随机生成，每次显示结果并非一致。

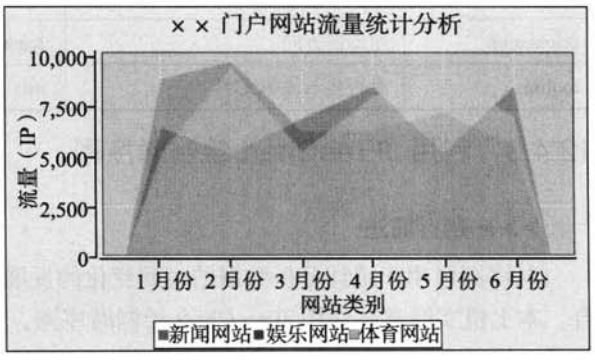


图 12.13　区域图

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

➤➤➤ 技术指导

在实现本实例时，区域图可以通过 JFreeChart 组件生成，生成此图主要分为以下两步来完成。

（1）创建生成区域图的数据集合

CategoryDataset 接口是 JFreeChart 组件提供一个数据集合接口，通过它的实现类 DefaultCategoryDataset 类可以创建区域图的数据集合，其方法为 addValue()，语法格式如下：

```
public void addValue(Number value, Comparable rowKey,Comparable columnKey)
```

- value：数据的键值。
- rowKey：行数据的键名。
- columnKey：列数据的键名。

（2）创建 JFreeChart 对象

本实例使用 ChartFactory 工厂类的 createAreaChart() 静态方法创建用于绘制区域图的制图对象，其语法格式如下：

```
public static JFreeChart createAreaChart(String title,
        String categoryAxisLabel,
        String valueAxisLabel,
        CategoryDataset dataset,
        PlotOrientation orientation,
        boolean legend,
        boolean tooltips,
        boolean urls)
```

createAreaChart() 方法的参数说明如表 12.15 所示。

表 12.15　createAreaChart() 方法的参数说明

参　数	说　明	参　数	说　明
title	图表的标题	categoryAxisLabel	横轴上的标签文字
valueAxisLabel	纵轴上的标签文字	dataset	制图所需的数据集合
orientation	绘图的方向	legend	是否显示图例
tooltips	是否显示说明文字	urls	是否生成超链接

12.4.3　利用 JFreeChart 绘制时序图

➤➤➤ 题目描述

时序图可用于统计分析数据的时间变化的发展情况，它的创建需要用到 XYDataset 数据集合。本上机实践尝试使用 JFreeChart 绘制时序图，首先使用 TimeSeries 对象填充一年 365 天的数据，然后通过时序图统计产品的年销量，运行结果如图 12.14 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

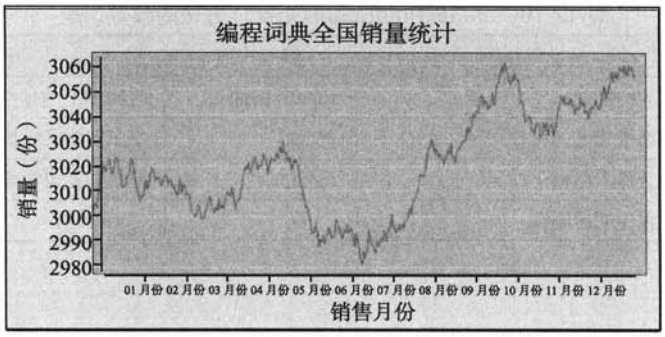


图 12.14　时序图

说明　本实例要求数据集合的数据随机生成，每次显示结果并非一致。

➤➤➤ 技术指导

在实现本实例时，时序图可以通过 JFreeChart 组件生成，生成此图主要分为以下两步来完成。

（1）创建生成时序图的数据集合

TimeSeries 类是描述时间序列的一个数据集对象，该对象的 add() 方法可以为数据集添加新的数据项，其语法格式如下：

```
public void add(RegularTimePeriod period, double value)
```

❑ period：添加到数据集的键名，它是一个日期或时间对象。

❑ value：添加到数据集的键值。

当构建了时间序列数据集对象后，通过 TimeSeriesCollection 类创建绘制时序图所用的数据集合，此类是 XYDataSet 的实现类，其语法格式如下：

```
public TimeSeriesCollection(TimeSeries series)
```

❑ series：时间序列数据集对象。

（2）根据数据集合生成 JFreeChart 对象

JFreeChart 提供了制图工厂类，该类的 createTimeSeriesChart() 静态方法用于创建时序图的制图对象，其语法格式如下：

```
public static JFreeChart createTimeSeriesChart(String title,
                                                String timeAxisLabel,
                                                String valueAxisLabel,
                                                XYDataset dataset,
                                                boolean legend,
                                                boolean tooltips,
                                            boolean urls)
```

createTimeSeriesChart() 方法的参数说明如表 12.16 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 12.16 createTimeSeriesChart() 方法的参数说明

参 数	说 明	参 数	说 明
title	图表的标题	timeAxisLabel	时间轴上的标签文字
valueAxisLabel	数据值轴上的标签文字	dataset	制图所需的数据集合
legend	是否显示图例	tooltips	是否显示说明文字
urls	是否生成超链接		

12.5 小结

本章向读者介绍开源组件 JFreeChart 的使用方法，学习本章内容，重点要掌握 JFreeChart 组件的核心对象。在使用过程中需要注意核心对象中子类与父类之间的关系，从而确定在什么情况使用何种对象。随着 JFreeChart 组件版本的不断更新，它的功能也越来越强大，其制图效果也越来越完善。JFreeChart 组件所能绘制的图表是千变万化的，本章所涉及的图表均为常用图形报表，主要是对其方法进行讲解，其更多应用请查阅 JFreeChart 组件的演示程序及 API 文档资料。

12.6 习题

选择题

1. 设置饼图数据区分类标签的字体颜色，需要通过 PiePlot 对象的（ ）设置。
 A. setLabelFont(Font font) 方法　　B. setLabelPaint(Paint paint) 方法
 C. setLabelLinkPaint(Paint paint) 方法
2. 在通过 ChartFactory 绘图工厂对象创建时序图时，调用的方法是（ ）。
 A. createBarChart()　　B. createPieChart()
 C. createBarChart3D()　　D. createTimeSeriesChart()
3. 以下哪种图表，可以更贴切地描述某种股票一个月内的走势情况（ ）。
 A. 饼状图表　　B. 柱状图表　　C. 时序图表　　D. 区域图表
4. 利用 AbstractRenderer 图形渲染对象，指定分类图形颜色的方法是（ ）。
 A. setSeriesPaint()　　B. setSeriesShape()　　C. setPaint()　　D. setStroke()
5. 利用 AbstractRenderer 图形渲染对象，指定图形边框的线条颜色的方法是（ ）。
 A. setOutLineStroke ()　　B. setSeriesOutLinePaint ()
 C. setSeriesOutLineStroke ()　　D. setOutLinePaint ()

填空题

1. 创建饼状图表时，需要创建的数据集合类型为_____。
2. 创建柱状图表时，需要创建的数据集合类型为_____。
3. 创建区域图表时，应用的是制图工厂对象 ChartFactory 的_____方法。
4. 获取时序图的 X 轴对象，需要调用绘图区对象 XYPlot 的_____方法。
5. 在创建饼状图时，如果数据集合中的数据来源于数据库，那么数据集合类型为_____。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 13 章　在 JSP 中操作 XML

视频讲解：29 分钟

XML 是现在比较流行的一种技术，它适用于不同应用程序间的数据交换，而且这种交换不以预先定义的一组数据结构为前提，增强了可扩展性。同时，在应用 Ajax 开发网站时，XMLHttpRequest 对象与服务器交换的数据，通常也采用 XML 格式。而在 JSP 中，为了方便快捷地操作 XML，通常需要使用专门用来解析 XML 的组件。dom4j 是一种解析 XML 文档的开源组件，它是目前性能最好的 XML 文档解析组件，许多开源项目中都采用 dom4j。因此，熟练掌握 XML 以及应用 dom4j 操作 XML，对于网站开发人员非常重要。本章将首先对 XML 进行简要介绍，然后再详细介绍应用 dom4j 操作 XML。

通过学习本章，读者可以达到以下学习目的：

- 了解 XML 的文档结构；
- 了解 XML 的语法要求；
- 掌握如何为 XML 文档中的元素定义属性；
- 了解 XML 的注释；
- 掌握 XML 中处理字符数据的两种方法；
- 掌握应用 dom4j 创建 XML 文件的方法；
- 掌握应用 dom4j 解析 XML 文档的方法；
- 掌握应用 dom4j 修改 XML 文档的方法。

13.1　XML简介

XML 语言是 eXtensible Markup Language（可扩展标记语言）的缩写，是 SGML（标准通用化标记语言）的一个子集，用于提供数据描述格式，适用于不同应用程序间的数据交换，而且这种交换不以预先定义的一组数据结构为前提，增强了可扩展性。下面将对 XML 的基础知识进行介绍。

13.1.1　XML 文档结构

XML 是一套定义语义标记的规则，也是用来定义其他标识语言的元标识语言。使用 XML 时，首先要了解 XML 文档的基本结构，然后再根据该结构创建所需的 XML 文档。下面我们先通过一个简单的 XML 文档来说明 XML 文档的结构。placard.xml 文件的代码如下：

```
<? xml version="1.0"encoding="UTF-8"><!-- 说明是 XML 文档，并指字 XML 文档的版本和编码 -->
<placard version="2.0">                <!-- 定义 XML 文档的根元素，并设置 version 属性 -->
  <description> 公告栏 </description>    <!-- 定义 XML 文档元素 -->
  <createTime> 创建于 2012 年 07 月 17 日 16 时 09 分 </createTime>
  <info id="1">                          <!-- 定义 XML 文档元素 -->
    <title> 重要通知 </title>
    <content>![CDATA[ 今天下午 4:50 将进行乒乓球比赛，请各位选手做好准备。 ]]></content>
    <pubDate>2012-07-17 16:09:37</pubDate>
</info>                                  <!-- 定义 XML 文档元素的结束标记 -->
<info id ="2">
    <title> 心中的鹅卵石 </title>
    <content>![CDATA[ 心中的鹅卵石 ]]></content>
```

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    <pubDate>2012-07-17 16:15:33</pubDate>
  </info>
</placard>                              <!-- 定义 XML 文档的根元素的结束标记 -->
```

一个基本的 XML 文档通常由序言和文档元素两部分组成，下面分别进行介绍。

1. 序言

XML 文档的序言可以包括 XML 声明、处理指令和注释，但这 3 项不是必需的，例如，在上面的文档中，就没有包括处理指令。

XML 文档的第一行通常是 XML 文档的声明，用于说明这是一个 XML 文档。XML 文档的声明并不是必需的，但通常建议为 XML 文档增加 XML 文档声明。XML 声明的语法格式如下：

```
<?xml version="version" encoding="value" standalone="value"?>
```

- version：用于指定遵循 XML 规范的版本号。在 XML 声明中必须包含 version 属性，该属性必须放在 XML 声明中其他属性之前。
- encoding：用于指定 XML 文档中字符使用的编码集。常用的编码集为 GBK 或 GB2312（简体中文）、BIG5（繁体中文）、ISO-8859-1（西欧字符）和 UTF-8（通用的国际编码）。

注意 如果在 XML 文档中没有指定编码集，那么该 XML 文档将不支持中文。

- standalone：用于指定该 XML 文档是否和一个外部文档嵌套使用。取值为 yes 或者 no，设置属性值为 yes 说明是一个独立的 XML 文档，与外部文件无关联，设置属性值为 no，说明 XML 文档不独立。

2. 文档元素

XML 文档中的元素是以树型分层结构排列的，一个元素可以嵌套在另一个元素中。XML 文档中有且只有一个顶层元素，称为文档元素或者根元素，类似于 HTML 页中的 <body> 元素，其他所有元素都嵌套在根元素中。

XML 文档元素由起始标记、元素内容和结束标记 3 部分组成。定义 XML 文档元素的语法格式如下：

```
<TagName>content</TagName>
```

- <TagName>：是 XML 文档元素的起始标记，其中 TagName 是元素的名字，具体的命名规则如下：
 - 元素的名字可以包含字母，数字和其他字符，但最好不使用“-”和“.”以免产生混淆。
 - 元素的名字只能以字母、下划线“_”或冒号“：”开头。
 - 元素的名字不能以 XML（包括 xml、Xml、xMl……）开头。
 - 元素的名字中不能包含空格。
 - 元素的名字不能为空，至少含有一个字母。
- content：是元素内容，可以包含其他的元素、字符数据、字符引用、实体引用、处理命令、注释和 CDATA 部分。
- </TagName>：是 XML 元素的结束标记，其中 TagName 是元素的名字，该名称必须与起始标记中指定的元素名称相同，包括字母的大小写。

说明 在本节开头处给出的代码中，placard 为根元素，info 为根元素的子元素。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

13.1.2　XML 语法要求

了解了 XML 文档的基本结构后，接下来还需要熟悉创建 XML 文档的语法要求。创建 XML 文档的语法要求如下：

1）XML 文档必须有一个顶层元素，其他元素必须嵌入在顶层元素中。

2）元素嵌套要正确，不允许元素间相互重叠或跨越。

3）每一个元素必须同时拥有起始标记和结束标记。这点与 HTML 不同，XML 不允许忽略结束标记。

4）起始标记中的元素类型名必须与相应结束标记中的名称完全匹配。

5）XML 元素类型名区分大小写，而且开始和结束标记必须准确匹配。例如，分别定义起始标记 <Title>、结束标记 </title>，由于起始标记的类型名与结束标记的类型名不匹配，说明元素是非法的。

6）元素类型名称中可以包含字母、数字以及其他字母元素类型，也可以使用非英文字符。

7）元素可以包含属性，但属性值必须用单引号或双引号括起来，前后两个引号必须一致，不能一个是单引号，一个是双引号。在一个元素节点中，属性名不能重复。

13.1.3　为 XML 文档中的元素定义属性

在一个元素的起始标记中，可以自定义一个或者多个属性。属性是依附于元素存在的。属性值用单引号或者双引号括起来。

例如，给元素 info 定义属性 id，用于说明公告信息的 ID 号。

```
<info id="1">
```

给元素添加属性是为元素提供信息的一种方法。当使用 CSS 样式表显示 XML 文档时，浏览器不会显示属性以及其属性值。若使用数据绑定、HTML 页中的脚本或者 XSL 样式表显示 XML 文档则可以访问属性及属性值。

注意　相同的属性名不能在元素起始标记中出现多次。

13.1.4　XML 的注释

注释是为了便于阅读和理解而在 XML 文档添加的附加信息。注释是对文档结构或者内容的解释，不属于 XML 文档的内容，所以 XML 解析器不会处理注释内容。XML 文档的注释以字符串“<!--”开始，以字符串“-->”结束。XML 解析器将忽略注释中的所有内容，这样可以在 XML 文档中添加注释说明文档的用途，或者临时注释掉没有准备好的文档部分。

注意　在 XML 文档中，解析器将“-->”看做是一个注释结束符号，所以字符串“-->”不能出现在注释的内容中，只能作为注释的结束符号。

13.1.5　处理字符数据

在 XML 文档中，有些字符会被 XML 解析器当做标记进行处理。如果希望把这些字符作为普通字符处理，就需要使用实体引用或 CDATA 段。下面进行详细介绍。

1．使用实体引用

为了避免系统将字符串中的特殊字符当成 XML 保留字符，XML 提供了一些实体引用。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在字符串中需要使用这些特殊字符时，就可以使用这些实体引用。XML 常用的实体引用如表 13.1 所示。

表 13.1　XML 常用的实体引用

字　符	实体引用	字　符	实体引用	字　符	实体引用
< 小于	<	> 大于	>	& 和	&
' 单引号	'	" 双引号	"		

例如，下面的代码，在浏览器中运行时，将显示如图 13.1 所示的错误提示。这是因为文档中出现了字符 &。

```
<?xml version="1.0" encoding="UTF-8"?><!-- 说明是 XML 文档，并指定 XML 文档的版本和编码 -->
<placard version="2.0">          <!-- 定义 XML 文档的根元素，并设置 version 属性 -->
  <info id="1">
    <title> 重要通知 </title>
    <content>   明天下午 3 点将举行乒乓球比赛的颁奖仪式！ </content>
    <pubDate>2012-06-21 16:20:48</pubDate>
  </info>
</placard>                                  <!-- 定义 XML 文档的根元素的结束标记 -->
```



图 13.1　未使用实体引用时的运行结果

将上面的代码中的 & 修改为“&”，在浏览器中运行将显示如图 13.2 所示的结果。修改后的代码如下：

```
<?xml version="1.0" encoding="UTF-8"?><!-- 说明是 XML 文档，并指定 XML 文档的版本和编码 -->
<placard version="2.0">                  <!-- 定义 XML 文档的根元素，并设置 version 属性 -->
  <info id="1">
    <title> 重要通知 </title>
    <content>&amp;nbsp;&amp;nbsp; 明天下午 3 点将举行乒乓球比赛的颁奖仪式！ </content>
    <pubDate>2012-06-21 16:20:48</pubDate>
  </info>
</placard>                                  <!-- 定义 XML 文档的根元素的结束标记 -->
```



图 13.2　应用实体引用的运行结果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2．使用 CDATA 段

CDATA 段是一种用来包含文本的方法，它内部的所有内容都会被 XML 解析器当做普通文本，所以任何符号都不会被认为是标记符。在 CDATA 标记下，实体引用将失去作用。CDATA 的语法格式如下：

```
<![CDATA[ 文本内容 ]]>
```

注意 CDATA 段不能进行嵌套，即 CDATA 段中不能再包含 CDATA 段。另外在字符串“]]>”之间不能有空格或换行符。

例如，在下面的 XML 文档中，由于 content 元素中包含的特殊字符比较多，使用实体引用比较麻烦，所以就需要使用 CDATA 段将 content 元素的内容括起来。

```
<?xml version="1.0" encoding="UTF-8"?>  <!-- 说明是 XML 文档，并指定 XML 文档的版本和编码 -->
<placard version="2.0">                 <!-- 定义 XML 文档的根元素，并设置 version 属性 -->
  <info id="1">                         <!-- 定义 XML 文档元素 -->
    <title>在 servlet 中弹出 JavaScript</title>
    <content><![CDATA[PrintWriter out = response.getWriter();
       out.println("<script>alert('修改成功！');</script>");]]></content>
    <pubDate>2012-06-15 16:12:06</pubDate>
  </info>
</placard>                                        <!-- 定义 XML 文档的根元素的结束标记 -->
```

上面代码在 IE 浏览器中的运行结果如图 13.3 所示。



图 13.3　使用 CDATA 段将 content 元素的内容括起来

13.2　dom4j概述

13.2.1　dom4j 简介

dom4j 是 sourceforge.net 上的一个 Java 开源项目，主要用于操作 XML 文档。例如，构造 XML 文档和解析 XML 文档。dom4j 应用于 Java 平台，采用了 Java 集合框架并完全支持 DOM、SAX 和 JAXP，是一种适合 Java 程序员使用的 Java XML 解析器。它具有性能优异、功能强大和易于使用等特点。目前，越来越多的 Java 软件都在使用 dom4j 来读写 XML。

13.2.2　dom4j 的下载与配置

在使用 dom4j 解析 XML 文档时，需要先下载 dom4j 组件。dom4j 组件可以到 http://sourceforge.net/projects/dom4j/ 网站中下载，具体的下载过程如下。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Step 1) 在浏览器的地址栏中输入 URL 地址 http://sourceforge.net/projects/dom4j/，并按下〈Enter〉键，进入到如图 13.4 所示的 dom4j 页面。



图 13.4　sourceforge.net 上的 dom4j 页面

Step 2) 在图 13.4 中，可以单击 Download 超链接，下载 1.6.1 版本的 dom4j 组件，该版本是一个比较稳定的版本，也可以单击 Browse All Files 超链接，进入到如图 13.5 所示的下载列表页面。



图 13.5　下载列表页面

Step 3) 单击 dom4j-2.0.0-ALPHA-2 超链接，将进入到如图 13.6 所示的选择下载文件的页面，在该页面中，提供了 3 个下载文件。

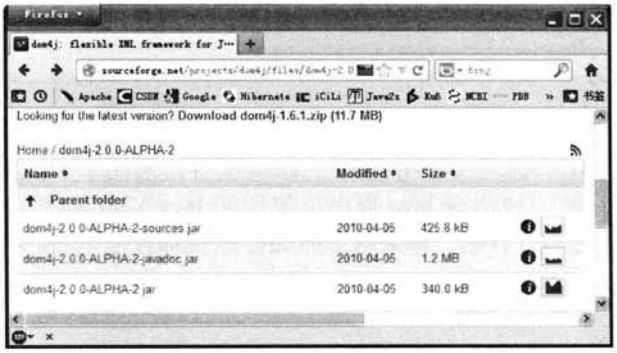


图 13.6　选择下载文件的页面

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Step 4 单击 dom4j-2.0.0-ALPHA-2.jar 超链接，将弹出如图 13.7 所示的下载页面，单击“保存文件”按钮，下载 dom4j-2.0.0-ALPHA-2.jar 文件。

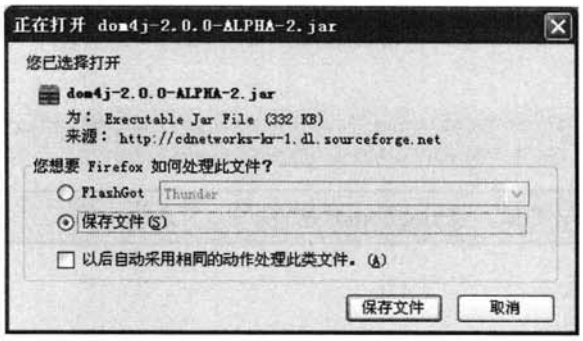


图 13.7　文件下载页面

Step 5 dom4j-2.0.0-ALPHA-2.jar 文件下载完成后，在需要应用 dom4j 组件的项目中，将 dom4j-2.0.0-ALPHA-2.jar 文件配置到构建路径中，或者将这个文件直接复制到项目的 lib 文件夹中。另外，dom4j 组件还需要 jaxen-1.1.4.jar 的支持，如果没有导入 jaxen-1.1.4.jar 文件，在执行时可能抛出“HTTP Status 500”异常，如图 13. 8 所示。

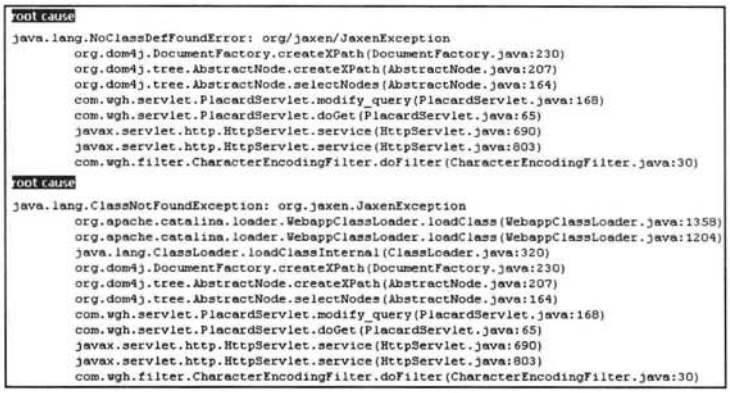


图 13. 8　缺少 jaxen-1.1.4.jar 时抛出的异常

说明　jaxen-1.1.4.jar 组件可以到它的官网 http://jaxen.codehaus.org/releases.html 中进行下载。

13.3　创建XML文件

dom4j 组件一个最重要的功能就是创建 XML 文件。通过 dom4j 组件可以很方便地创建 XML 文件。下面将详细介绍如何应用 dom4j 组件创建 XML 文件。

13.3.1　创建 XML 文档对象

使用 DocumentHelper 类的 createDocument() 方法可以创建一个 XML 文档对象。创建 XML 文档对象的具体代码如下：

```
Document document = DocumentHelper.createDocument();
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明 DocumentHelper 类保存在 org.dom4j 包中。

另外，使用 DocumentFactory 对象也可以创建一个 XML 文档对象。DocumentFactory 对象由 DocumentFactory 类的 getInstance() 静态方法产生。通过 DocumentFactory 对象创建 XML 文档对象的具体代码如下：

```
DocumentFactory documentFactory= DocumentFactory.getInstance();
Document document=documentFactory.createDocument();
```

说明 DocumentFactory 类保存在 org.dom4j 包中。

13.3.2 创建根节点

为 XML 文档创建根节点，首先需要创建一个普通节点，然后再通过调用 Document 对象的 setRootElement() 方法把该节点设置为根节点。创建普通节点可以通过 DocumentHelper 对象的 createElement() 方法实现。下面将对 DocumentHelper 对象的 createElement() 方法和 Document 对象的 setRootElement() 方法进行详细介绍。

（1）DocumentHelper 对象的 createElement() 方法

DocumentHelper 对象的 createElement() 方法用于创建一个普通的节点，其方法原型如下：

```
public static Element createElement(String name)
```

❑ name：用于指定要创建的节点名。

（2）Document 对象的 setRootElement() 方法

Document 对象的 setRootElement() 方法用于将指定的节点设置为根节点，其方法原型如下：

```
public void setRootElement(Element rootElement)
```

❑ rootElement：用于指定要作为根节点的普通节点。

例如，创建一个只包括一个根节点的 XML 文档的具体代码如下：

```
Document document = DocumentHelper.createDocument();       // 创建 XML 文档对象
Element placard=DocumentHelper.createElement("placard"); // 创建普通节点
document.setRootElement(placard);                          // 将 placard 设置为根节点
```

13.3.3 添加注释

在创建 XML 文档时，为了便于阅读和理解，经常需要在 XML 文档中添加注释。通过 dom4j 组件的 Element 对象的 addComment() 方法可以为指定的节点添加注释。Element 对象的 addComment() 方法的原型如下：

```
public Element addComment(String comment)
```

❑ comment：用于指定注释内容。

例如，为 XML 文档的根节点添加注释的代码如下：

```
Document document = DocumentHelper.createDocument();       // 创建 XML 文档对象
Element placard=DocumentHelper.createElement("placard"); // 创建普通节点
document.setRootElement(placard);                          // 将 placard 设置为根节点
placard.addComment("这是根节点");                          // 添加注释
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

13.3.4　添加属性

在创建 XML 文档时，经常需要为指定的节点添加属性。通过 dom4j 组件的 Element 对象的 addAttribute() 方法可以为指定的节点添加属性。Element 对象的 addAttribute() 方法的原型如下：

```
public Element addAttribute(String name,String value)
```

- name：用于指定属性名。
- value：用于指定属性值。

例如，为 XML 文档的根节点添加 version 属性的代码如下：

```
Document document = DocumentHelper.createDocument();      // 创建 XML 文档对象
Element placard=DocumentHelper.createElement("placard"); // 创建普通节点
document.setRootElement(placard);                         // 将 placard 设置为根节点
placard.addAttribute("version", "2.0");                   // 添加属性
```

13.3.5　创建子节点

在创建 XML 文档时，经常需要为指定的节点添加子节点，例如，为根节点添加子节点。通过 dom4j 组件的 Element 对象的 addElement() 方法可以为指定的节点添加子节点。Element 对象的 addElement() 方法的原型如下：

```
public Element addElement(String name)
```

- name：用于指定子节点的名称。

说明　Element 对象的 addElement() 方法是从 org.dom4j.Branch 接口中继承的。

例如，为根节点 placard 创建一个子节点 description 的具体代码如下：

```
Document document = DocumentHelper.createDocument();      // 创建 XML 文档对象
Element placard=DocumentHelper.createElement("placard"); // 创建普通节点
document.setRootElement(placard);                         // 将 placard 设置为根节点
Element description = placard.addElement("description"); // 创建子节点 description
```

13.3.6　设置节点的内容

创建节点后，还需要为节点设置内容，在 dom4j 中提供了两种设置节点内容的方法，一种是将节点内容设置为普通文本，另一种是将节点内容设置为 CDATA 段。下面分别进行介绍。

（1）将普通文本作为节点内容

将普通文本作为节点内容可以通过 dom4j 组件的 Element 对象的 setText() 方法进行设置。Element 对象的 setText() 方法的原型如下：

```
public void setText(String text)
```

- text：用于指定节点内容。

说明　Element 对象的 setText() 方法是从 org.dom4j.Node 接口中继承的。

例如，设置子节点 description 的内容为“公告栏”的代码如下：

```
Element description = placard.addElement("description");      // 创建子节点
description.setText("公告栏");                                 // 设置子节点的内容
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（2）将 CDATA 段作为节点内容

将 CDATA 段作为节点内容可以通过 dom4j 组件的 Element 对象的 addCDATA() 方法进行设置。Element 对象的 addCDATA () 方法的原型如下：

```
public Element addCDATA(String cdata)
```

❑ cdata：用于指定 CDATA 段中的文本内容。

例如，设置子节点 content_item 的内容的 CDATA 段代码如下：

```
Element content_item = placard.addElement("content");    //创建子节点
content_item.addCDATA("心中的鹅卵石 & 童年的梦");          //设置子节点的内容
```

13.3.7 设置编码

在应用 dom4j 创建 XML 文档时，默认的编码集为 UTF-8，但有时不一定要使用该编码集，这时就可以通过 dom4j 的 OutputFormat 类提供的 setEncoding() 方法设置文档的编码集。setEncoding() 方法的原型如下：

```
public void setEncoding(String encoding)
```

encoding：用于指定编码集。常用的编码集为 GBK 或 GB2312（简体中文）、BIG5（繁体中文）、ISO-8859-1（西欧字符）和 UTF-8（通用的国际编码）。

说明 OutputFormat 类保存在 org.dom4j.io 包中。

例如，设置 XML 文档的编码为 GBK 的具体代码如下：

```
OutputFormat format=new OutputFormat();          //创建 OutputFormat 对象
format.setEncoding("GBK");                       //设置写入流编码为 GBK
```

应用上面代码后，将在生成的 XML 文档的声明中，将编码集设置为 GBK，设置完成的代码如下：

```
<?xml version="1.0" encoding="GBK" ?>
```

13.3.8 设置输出格式

应用 dom4j 生成 XML 文件时，在默认的情况下，生成的 XML 文件采用紧凑方式排版，采用这种排版格式比较混乱，不容易阅读。为此，dom4j 提供了将输出格式设置为缩进方式的方法，这样可以美化 XML 文件，使 XML 文件更容易阅读。应用 dom4j 的 OutputFormat 类提供的 createPrettyPrint() 方法可以将 XML 文件格式化为缩进方式。createPrettyPrint() 方法的原型如下：

```
public static OutputFormat createPrettyPrint()
```

createPrettyPrint() 方法无入口参数，返回值为 OutputFormat 对象。例如，将 XML 文档设置为缩进方式的具体代码如下：

```
OutputFormat format = OutputFormat.createPrettyPrint();          // 格式化为缩进方式
```

13.3.9 输出 XML 文件

在 XML 文档对象创建完成，并添加相应的节点后，还需要输出该 XML 文档，否则用户

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



将不能看到 XML 的内容。在应用 dom4j 创建 XML 文档时，有以下两种输出方式。

（1）未设置输出格式

当没有为 XML 文档设置输出格式时，可以使用 XMLWriter 类的构造方法 XMLWriter (Writer writer) 实例化一个 XMLWriter 对象，再应用该对象的 write() 方法写入数据，最后关闭 XMLWriter 对象。

例如，将 XML 文档对象 document 写入到 XML 文件的具体代码如下：

```
String fileURL = request.getRealPath("/xml/placard.xml");
XMLWriter writer = new XMLWriter(new FileWriter(fileURL));  // 实例化 XMLWriter 对象
writer.write(document);                                     // 向流写入数据
writer.close();                                             // 关闭 XMLWriter
```

（2）已经设置了输出格式或编码集

当已经为 XML 文档设置输出格式或编码集时，可以使用 XMLWriter 类的构造方法 XMLWriter(Writer writer, OutputFormat format) 实例化一个 XMLWriter 对象，再应用该对象的 write() 方法写入数据，最后关闭 XMLWriter 对象。

例如，将设置编码集为 GBK 的 XML 文档对象 document 写入到 XML 文件的具体代码如下：

```
OutputFormat format=new OutputFormat();                    // 创建 OutputFormat 对象
format.setEncoding("GBK");                                 // 设置写入流编码
String fileURL = request.getRealPath("/xml/placard.xml");
                                                           // 实例化 XMLWriter 对象
XMLWriter writer = new XMLWriter(new FileWriter(fileURL), format);
writer.write(document);                                    // 向流写入数据
writer.close();                                            // 关闭 XMLWriter
```

技巧　如果不想将 XML 文档输出到 XML 文件中，可以应用以下代码将其输出到浏览器或控制台上。

```
/*********************** 将其输出到控制台的代码 **********************************/
XMLWriter writer = new XMLWriter(System.out,format);
writer.write(document);
/*********************** 将其输出到浏览器的代码 **********************************/
XMLWriter writer= new XMLWriter(out,format);
writer.write(document);
```

在将 XML 文档输出到浏览器的代码中，out 为 java.io.PrintWriter 对象，可以通过 response 对象的 response.getWriter() 方法获得，out 也可以是 JSP 的内置对象。

注意　在将 XML 文档输出到控制台时，不能调用 XMLWriter 对象的 close() 方法。

【例 13.1】 应用 dom4j 组件创建显示图书信息的 XML 文档，并显示到浏览器上。

实例位置：光盘 \MR\ 源码 \ 第 13 章 \13.1

Step 1）创建动态 Web 项目，并将 dom4j-2.0.0-ALPHA-2.jar 和 jaxen-1.1.4.jar 文件添加到 Web 应用的 WEB-INF\lib 文件夹中。

Step 2）编写并配置 Servlet，在 doPost() 方法编写以下代码，创建显示图书信息的 XML 文档，并将其输出到 XML 文件中。

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            throws ServletException, IOException {
        response.setContentType("text/xml");
        Document document = DocumentHelper.createDocument();  // 创建 Document 对象
        Element root = DocumentHelper.createElement("Book-info");
                                                               // 创建根节点
        document.setRootElement(root);
        Element book1 = root.addElement("book");               // 创建图书节点
        book1.addAttribute("名称", "Java 开发典型模块大全");
        Element book1_price = book1.addElement("价格");// 创建图书节点的子节点
        book1_price.setText("79");
        Element book1_author = book1.addElement("作者");// 创建图书节点的子节点
        book1_author.setText("明日科技");
        Element book2 = root.addElement("book");               // 创建图书节点
        book2.addAttribute("名称", "JAVA 从入门到精通");
        Element book2_price = book2.addElement("价格");// 创建图书节点的子节点
        book2_price.setText("59.8");
        Element book2_author = book2.addElement("作者");// 创建图书节点的子节点
        book2_author.setText("明日科技");
        OutputStream out = response.getOutputStream();         // 获取输出流
        XMLWriter xmlWriter = new XMLWriter(out);              // 创建 XMLWriter
        xmlWriter.write(document);                             // 输出 XML 文档
        xmlWriter.flush();
        xmlWriter.close();
    }
```

运行本实例，将显示如图 13.9 所示的运行结果。

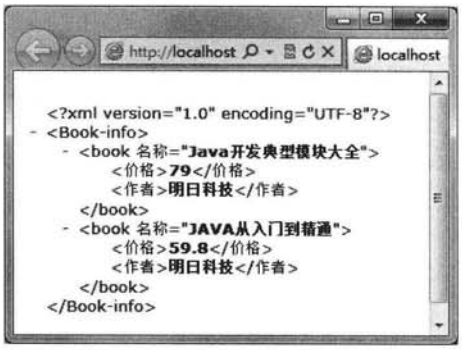


图 13.9　输出的显示图书信息的文档

13.4　解析XML文档

dom4j 组件的另一个最重要的功能就是解析 XML 文档。通过 dom4j 组件可以很方便地解析 XML 文档。下面将详细介绍如何应用 dom4j 组件解析 XML 文档。

13.4.1　构建 XML 文档对象

在解析 XML 文档前，需要构建要解析的 XML 文件所对应的 XML 文档对象。在获取 XML 文档对象时，首先需要创建 SAXReader 对象，然后调用该对象的 read() 方法获取对应的 XML 文档对象。SAXReader 对象的 read() 方法的原型如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
public Document read(File file)
                throws DocumentException
```

❑ file：用于指定要解析的 XML 文件。

例如，获取 XML 文件 placard.xml 对应的 XML 文档对象的代码如下：

```
String fileURL = request.getRealPath("/xml/placard.xml");
SAXReader reader = new SAXReader();                    // 实例化 SAXReader 对象
Document document = reader.read(new File(fileURL));// 获取 XML 文件对应的 XML 文档对象
```

说明 SAXReader 类位于 org.dom4j.io 包中。

13.4.2　获取根节点

在构建 XML 文档对象后，就可以通过该 XML 文档对象获取根节点。dom4j 组件的 Document 对象的 getRootElement() 方法可以返回指定 XML 文档的根节点。getRootElement() 方法的原型如下：

```
public Element getRootElement()
```

❑ 返回值：Element 对象。

例如，获取 XML 文档对象 document 的根节点的代码如下：

```
Element placard = document.getRootElement();          // 获取根节点
```

13.4.3　获取子节点

在获取根节点后，还可以获取其子节点，这可以通过 Element 对象的 element() 或 elements() 方法实现。下面将分别介绍这两个方法。

（1）element() 方法

element() 方法用于获取指定名称的第一个节点。该方法通常用于获取根节点中节点名唯一的一个子节点。element() 方法的原型如下：

```
public Element element(String name)
```

❑ name：用于指定要获取的节点名。

❑ 返回值：Element 对象。

（2）elements() 方法

elements() 方法用于获取指定名称的全部节点。该方法通常用于获取根节点中多个并列的具有相同名称的子节点。elements() 方法的原型如下：

```
public List elements(String name)
```

❑ name：用于指定要获取的节点名。

❑ 返回值：List 集合。

例如，保存公告信息的 XML 文件 placard.xml，具体代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<?xml version="1.0" encoding="UTF-8"?>   <!-- 说明是 XML 文档 -->
<placard version="2.0">                  <!-- 定义 XML 文档的根元素 -->
  <description> 公告栏 </description>      <!-- 定义 XML 文档元素 -->
  <createTime> 创建于 2012 年 07 月 25 日 16 时 09 分 </createTime>
  <info id="1">                                  <!-- 定义 XML 文档元素 -->
    <title > 重要通知 </title>
    <content><![CDATA[ 今天下午 4:50 将进行乒乓球比赛，请各位选手做好准备。 ]]></content>
    <pubDate>2012-07-25 16:09:37</pubDate>
  </info>                                <!-- 定义 XML 文档元素的结束标记 -->
  <info id="2">
    <title> 心中的鹅卵石 </title>
    <content><![CDATA[ 心中的鹅卵石 ]]></content>
    <pubDate>2012-07-25 16:15:33</pubDate>
  </info>
</placard>
```

要获取 description 节点，可以应用 element() 方法，具体代码如下：

```
Element root = document.getRootElement();                    // 获取根节点
Element description=root.element("description");             // 获取 description 节点
```

要获取 info 节点，可以应用 elements() 方法，具体代码如下：

```
Element root = document.getRootElement();                    // 获取根节点
List list_item = root.elements("info");                      // 获取 info 节点
```

13.5 修改XML文档

在对 XML 文档进行操作时，经常需要对 XML 文档进行修改。对 XML 文档进行修改通常包括修改节点和删除节点。下面将详细介绍应用 dom4j 组件修改和删除 XML 文档的节点的方法。

13.5.1 修改节点

在修改 XML 文档的节点前，首先需要查询到该节点。在 dom4j 组件中，查询节点可以应用 Element 对象的 selectSingleNode() 方法或 selectNodes() 方法实现。下面对这两个方法进行详细介绍。

（1）selectSingleNode() 方法

Element 对象的 selectSingleNode() 方法用于获取符合指定条件的唯一节点，该方法的原型如下：

```
public Node selectSingleNode(String xpathExpression)
```

- xpathExpression：XPath 表达式。XPath 表达式使用反斜杠 “/” 隔开节点树中的父子节点，从而构成代表节点位置的路径。如果 XPath 表达式以反斜杠 “/” 开头，则表示使用的是绝对路径，否则表示使用的是相对路径。如果使用属性，那么必须在属性名前加上 @ 符号。另外，在 XPath 表达式中，也可以使用谓词，例如下面的表达式将返回 ID 属性值等于 1 的 info 节点。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
/placard/info[@id='1']
```

说明　Element 对象的 selectSingleNode () 方法是从 org.dom4j.Node 接口中继承的。

例如，应用 selectSingleNode() 方法获取 XML 文档的根节点 placard 的 ID 属性为 1 的子节点 info 的代码如下：

```
org.dom4j.Node item=placard.selectSingleNode("/placard/info[@id='1]");
```

（2）selectNodes() 方法

Element 对象的 selectNodes() 方法则可以获取符合指定条件的节点列表，该方法的原型如下：

```
public List selectNodes(String xpathExpression)
```

❑ xpathExpression：XPath 表达式。

说明　Element 对象的 selectNodes() 方法是从 org.dom4j.Node 接口中继承的。

例如，应用 selectNodes() 方法获取 XML 文档的根节点 placard 的子节点 info 的代码如下：

```
List list = placard.selectNodes("/placard/info");
```

获取到要修改的节点后，就可以应用 13.3.3 小节、13.3.4 小节、13.3.5 小节和 13.3.6 小节介绍的方法为该节点添加注释、添加属性、添加子节点或设置节点内容等。

13.5.2　删除节点

在实现删除节点时，同样需要查询到要删除的节点，这可以应用 13.5.1 小节介绍的 Element 对象的 selectSingleNode() 方法或 selectNodes() 方法实现，这里不再赘述。获取到要删除的节点后，就可以应用 Element 对象的 remove() 方法删除该节点了。

```
public boolean remove(Element element)
```

❑ element：Element 对象。

❑ 返回值：true 或 false，表示节点是否删除成功。

说明　上面的 remove() 方法是从 org.dom4j.Branch 接口中继承的。

例如，要删除 XML 文档的根节点 placard 的 ID 属性为 1 的子节点 info 的代码如下：

```
Element item=(Element)placard.selectSingleNode("/placard/info[@id='1']");
                                                    // 获取要删除的节点
if (null != item) {
   placard.remove(item);                            // 删除指定节点
}
```

上面已经介绍了如何删除指定的一个节点，在实际编程中，有时还需要批量删除指定子节点。例如，要删除 XML 文档的根节点的子节点 info，可以使用以下代码：

```
document.getRootElement().elements("info").clear();      // 删除全部 info 节点
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

13.6 上机实践

13.6.1 保存公告信息到 XML 文件

➤➤➤ 题目描述

使用 dom4j 组件可以很方便地创建 XML 文件，并添加节点及节点内容。本练习尝试开发一个程序，要求使用 dom4j 组件将前台表单提交的公告信息保存到 XML 文件中。效果如图 13.10 和图 13.11 所示。

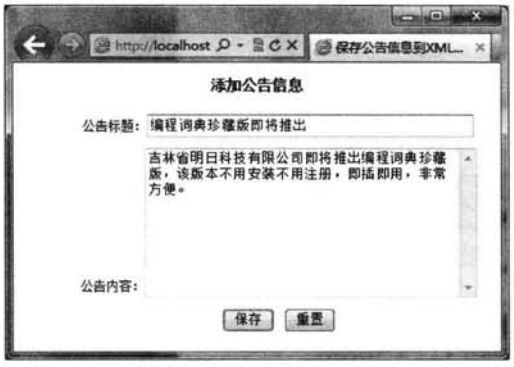


图 13.10 输入公告信息页面



图 13.11 打开的 XML 文件

➤➤➤ 技术指导

在实现本上机实践时，首先创建一个用户输入公告信息的 JSP 页面，然后创建一个将公告信息保存到 XML 文件的 Servlet，在该 Servlet 的 doPost() 方法中，使用 dom4j 组件将输入的公告信息保存到指定的 XML 文件中，如果该文件不存在，则创建它。本上机实践的关键代码如下：

```
response.setContentType("text/html");
String fileURL = request.getRealPath("/xml/placard.xml");//XML 文件的路径
File file = new File(fileURL);
String title=request.getParameter("title");          //获取公告标题
String content=request.getParameter("content");      //获取公告内容
Document document = null;
Element placard = null;
DateFormat df=new SimpleDateFormat("yyyy 年 MM 月 dd 日 HH 时 mm 分 ");
                                                     //设置日期格式
if (!file.exists()) {          // 判断文件是否存在，如果不存在，则创建该文件
       document = DocumentHelper.createDocument();
                                                     //创建 XML 文档对象
       placard=DocumentHelper.createElement("placard");
                                                     //创建普通节点
       document.setRootElement(placard);     // 将 placard 设置为根节点
       placard.addAttribute("version", "2.0");// 为根节点添加属性 version
       Element description = placard.addElement("description");
                                                     //添加 description 子节点
       description.setText(" 公告栏 ");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            Element createTime = placard.addElement("createTime");
                                                    //添加 createTime 子节点
            createTime.setText("创建于 "+df.format(new Date()));
    } else {
            SAXReader reader = new SAXReader();   //实例化 SAXReader 对象
            try {
                    document = reader.read(new File(fileURL));
                                                //获取 XML 文件对应的 XML 文档对象
                    placard = document.getRootElement();
                                                    //获取根节点
            } catch (DocumentException e) {
                    e.printStackTrace();
            }
    }
    /********************** 添加公告信息 **************************/
    String id = String.valueOf(placard.elements("info").size() + 1);
                                                    // 获取当前公告的 id 号
    Element info = placard.addElement("info");    //添加 info 节点
    info.addAttribute("id", id);                  //为 info 节点设置 ID 属性
    Element title_info = info.addElement("title");//添加 title 节点
    title_info.setText(title);                    //设置 title 节点的内容
    Element content_item = info.addElement("content");
                                                    //添加 content 节点
    // 此处不能使用 setText() 方法，如果使用该方法，当内容中出现 HTML 代码时，程序将出错
    content_item.addCDATA(content);       //设置 content 节点的内容为 CDATA 段
    Element pubDate_item = info.addElement("pubDate");
                                                    //添加 pubDate 节点
    df=new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");//设置日期格式
    pubDate_item.setText(df.format(new Date()));
    /************************ 保存文件 ************************************/
    // 创建 OutputFormat 对象
    OutputFormat format = OutputFormat.createPrettyPrint();
                                                    //格式化为缩进方式
    format.setEncoding("UTF-8");                  //设置写入流编码
    try {
            XMLWriter writer = new XMLWriter(new FileOutputStream(fileURL),
            format)
            writer.write(document);               // 向流写入数据
            writer.close();                       //关闭流
    } catch (IOException e) {
            e.printStackTrace();
    }
```

13.6.2　遍历 XML 文档

➤➤➤ 题目描述

使用 dom4j 组件不仅可以创建 XML 文件，而且可以修改或删除 XML 文件中的内容。本练习尝试开发一个程序，要求使用 dom4j 组件遍历 XML 文档，效果如图 13.12 和图 13.13 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 13.12　XML 文件的内容

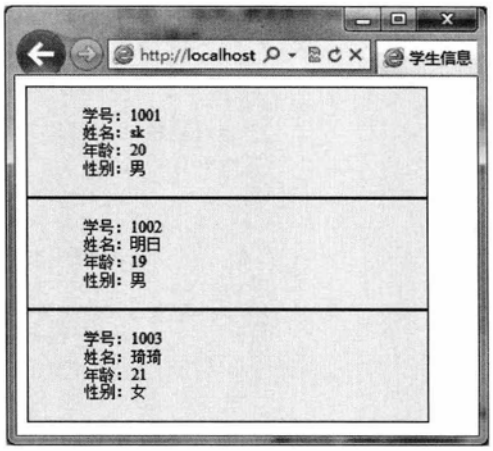


图 13.13　遍历后显示的结果

➤➤➤ 技术指导

在实现本上机实践时，主要是编写一个用于遍历 XML 文档的 Servlet，在该 Servlet 的 doPost() 方法中，使用 dom4j 组件获取 XML 文档中的学生子元素，并保存到 List 集合对象中，然后使用 for each 语句将该 List 集合中的学生信息转化为 Student 对象并保存到另一个 List 集合中。本上机实践的关键代码参考如下：

```
SAXReader reader = new SAXReader();
try {
        String file = getServletContext().getRealPath("/Student.xml");
                                        //获取要遍历 XML 文档的路径
        Document document = reader.read(new File(file));
                                        // 获取 XML 文件对应的 XML 文档对象
        Element root = document.getRootElement();
                                        //获取根元素
        List<Element> list = root.elements("student");
                                        //获取 student 子元素
        List<Student> students = new ArrayList<Student>();
                                        //创建用于保存学生信息的 List 集合对象
        //通过 for each 语句遍历获取到的 student 子元素
        for (Element element : list) {
                Student s = new Student();
                s.setId(Integer.valueOf(element.attributeValue("id")));
                                        //获取学号
                s.setName(element.elementText("name"));
                                        //获取姓名
                s.setSex(element.elementText("sex"));
                                        //获取性别
                s.setAge(Integer.valueOf(element.elementText("age")));
                                        //获取年龄
                students.add(s);
        }
        request.setAttribute("list", students);
} catch (Exception e) {
        e.printStackTrace();
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

13.6.3　管理保存在 XML 文件中的公告信息

➤➤➤ 题目描述

使用 dom4j 组件不仅可以创建 XML 文件，而且可以修改或删除 XML 文件中的内容。本练习尝试开发一个程序，要求使用 dom4j 组件修改或删除 XML 文件中指定节点的内容，效果如图 13.14 和图 13.15 所示。



图 13.14　公告列表页面



图 13.15　修改公告信息页面

➤➤➤ 技术指导

在实现本实例时，主要分为以下 3 个步骤实现。

Step 1) 读取 XML 文件中全部的公告信息。在实现时，首先获取 XML 文件对应的 Document 对象，然后获取描述信息及创建日期，再通过 for 循环获取公告列表，并保存到 List 集合中，最后将获取的描述信息、创建日期和公告列表保存到 HttpServletRequest 对象中。

Step 2) 修改公告的原内容。在实现时，首先需要获取到要修改的公告信息，然后再用获取的公告信息替换原公告信息，并保存。

Step 3) 删除指定的公告信息。在实现时，也需要先获取到要删除的公告信息，然后再使用 remove() 方法移除指定节点来删除。

13.7　小结

本章首先对 XML 文档结构、语法要求、属性、注释及如何处理字符数据进行简要介绍，然后对 dom4j 组件进行了简要介绍，接下来介绍了 dom4j 组件的下载与配置方法，最后详细介绍了如何应用 dom4j 组件创建 XML 文件、解析 XML 文档以及修改 XML 文档。其中，应用 dom4j 组件创建 XML 文件和解析 XML 文档需要读者重点掌握，这部分内容在进行 Ajax 开发时经常用到。另外，在为网站添加 RSS 订阅功能时，也可以通过 dom4j 创建 XML 文件和解析 XML 文档来实现。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

13.8 习题

选择题

1. XML 文档结构中不包括的内容为（　　）。
 A. XML 声明　　B. 注释　　C. Java 代码
2. 使用 dom4j 创建 XML 文档，使用的是（　　）类。
 A. DocumentHelper　　B. DocumentFactory　　C. Element
3. dom4j 中创建根节点使用的方法为（　　）。
 A. createElement 方法　　B. setRootElement 方法　　C. addComment 方法
4. dom4j 为 XML 文档添加注释使用的是（　　）。
 A. createElement 方法　　B. setRootElement 方法　　C. addComment 方法
5. dom4j 中为 XML 文件添加属性使用的是（　　）。
 A. createElement 方法　　B. setRootElement 方法　　C. addAttribute 方法

填空题

1. XML 文件结构包括_____方面内容。
2. dom4j 中应用_____表示文件对象。
3. 要获取 XML 文件中的根节点，需要使用_____方法。
4. 要使用 dom4j 删除节点，需要使用_____方法。
5. 要使用 dom4j 修改节点内容，需要使用_____方法。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 14 章　完美体验 Ajax 技术

视频讲解：34 分钟

随着 Web 2.0 概念的普及，追求更人性化、更美观的页面效果成了网站开发的必修课，Ajax 正在其中充当着重要角色。由于 Ajax 是一个客户端技术，所以无论使用哪种服务器端技术（如 JSP、PHP、ASP.NET 等）都可以使用 Ajax。相对于传统的 Web 应用开发，Ajax 运用的是更加先进、更加标准化、更加高效的 Web 开发技术体系。本章将介绍如何在 JSP 中应用 Ajax。

通过学习本章，读者可以达到以下学习目的：

- 认识 Ajax；
- 了解 Ajax 使用的技术；
- 掌握传统 Ajax 的工作流程；
- 掌握 jQuery 的基本使用方法；
- 掌握如何使用 jQuery 发送 GET 和 POST 请求；
- 了解使用 Ajax 需要注意的几个问题。

14.1　了解Ajax

14.1.1　什么是 Ajax

Ajax 是 Asynchronous JavaScript and XML 的缩写，意思是异步的 JavaScript 与 XML。Ajax 并不是一门新的语言或技术，它是 JavaScript、XML、CSS、DOM 等多种已有技术的组合，可以实现客户端的异步请求操作。从而可以实现在不需要刷新页面的情况下与服务器进行通信，减少了用户的等待时间，减轻了服务器和带宽的负担，提供更好的服务响应。

14.1.2　Ajax 开发模式与传统开发模式的比较

在 Web 2.0 时代以前，多数网站都采用传统的开发模式，而随着 Web 2.0 时代的到来，越来越多的网站都开始采用 Ajax 开发模式。为了让读者更好地了解 Ajax 开发模式，下面将对 Ajax 开发模式与传统开发模式进行比较。

在传统的 Web 应用模式中，页面中用户的每一次操作都将触发一次返回 Web 服务器的 HTTP 请求，服务器进行相应的处理（获得数据、运行与不同的系统会话）后，返回一个 HTML 页面给客户端。如图 14.1 所示。

而在 Ajax 应用中，页面中用户的操作将通过 Ajax 引擎与服务器端进行通信，然后将返回结果提交给客户端页面的 Ajax 引擎，再由 Ajax 引擎来决定将这些数据插入到页面的指定位置，如图 14.2 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

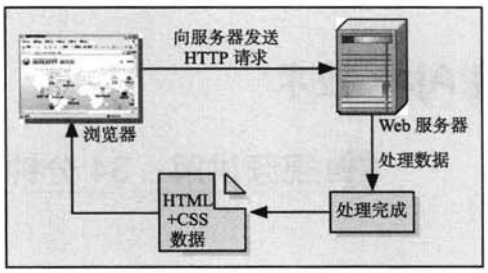


图 14.1　Web 应用的传统模型

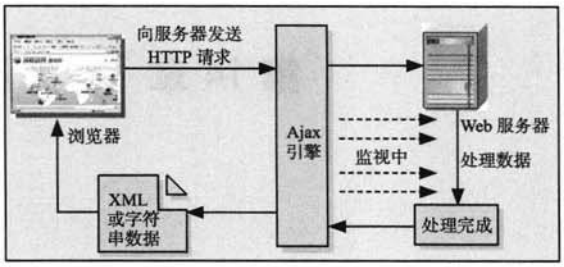


图 14.2　Web 应用的 Ajax 模型

从图 14.1 和图 14.2 中可以看出，对于每个用户的行为，在传统的 Web 应用模型中，将生成一次 HTTP 请求，而在 Ajax 应用开发模型中，将变成对 Ajax 引擎的一次 JavaScript 调用。在 Ajax 应用开发模型中通过 JavaScript 实现在不刷新整个页面的情况下，对部分数据进行更新，从而降低了网络流量，给用户带来了更好的体验。

14.1.3　Ajax 的优点

与传统的 Web 应用不同，Ajax 在用户与服务器之间引入一个中间媒介（Ajax 引擎），从而消除了网络交互过程中的“处理—等待—处理—等待”的缺点。使用 Ajax 的优点具体表现在以下几方面。

1）减轻服务器的负担。Ajax 的原则是“按需求获取数据”，这可以最大程度地减少冗余请求和响应对服务器造成的负担。

2）可以把一部分以前由服务器负担的工作转移到客户端，利用客户端闲置的资源进行处理，减轻服务器和带宽的负担，节约空间和成本。

3）无刷新更新页面，从而使用户不用再像以前那样在服务器处理数据时，只能在死板的白屏前焦急地等待。Ajax 使用 XMLHttpRequest 对象发送请求并得到服务器响应，在不需要重新载入整个页面的情况下，就可以通过 DOM 及时将更新的内容显示在页面上。

4）可以调用 XML 等外部数据，进一步促进页面显示和数据的分离。

5）基于标准化的并被广泛支持的技术，不需要下载插件或者小程序。

14.2　使用XMLHttpRequest对象

Ajax 是 XMLHttpRequest 对象和 JavaScript、XML、CSS、DOM 等多种技术的组合。其中，只有 XMLHttpRequest 对象是新技术，其他的均为已有技术。下面我们就对 XMLHttpRequest 对象进行详细介绍。

Ajax 使用的技术中，最核心的技术就是 XMLHttpRequest，它是一个具有应用程序接口的 JavaScript 对象，能够使用超文本传输协议（HTTP）连接一个服务器，是微软公司为了满足开发者的需要，于 1999 年在 IE 5.0 浏览器中率先推出的。通过 XMLHttpRequest 对象，Ajax 可以像桌面应用程序一样只同服务器进行数据层面的交换，而不用每次都刷新页面，也不用每次都将数据处理的工作交给服务器来完成，这样既减轻了服务器负担又加快了响应速度、缩短了用户等待的时间。

14.2.1　初始化 XMLHttpRequest 对象

在使用 XMLHttpRequest 对象发送请求和处理响应之前，首先需要初始化该对象，由于

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

XMLHttpRequest 不是一个 W3C 标准，所以对于不同的浏览器，初始化的方法也是不同的。通常情况下，初始化 XMLHttpRequest 对象只需要考虑两种情况，一种是 IE 浏览器，另一种是非 IE 浏览器，下面分别进行介绍。

（1）IE 浏览器

IE 浏览器把 XMLHttpRequest 实例化为一个 ActiveX 对象，具体方法如下：

```
var http_request = new ActiveXObject("Msxml2.XMLHTTP");
或者
var http_request = new ActiveXObject("Microsoft.XMLHTTP");
```

在上面的语法中，Msxml2.XMLHTTP 和 Microsoft.XMLHTTP 是针对 IE 浏览器的不同版本而进行设置的，目前比较常用的是这两种。

（2）非 IE 浏览器

非 IE 浏览器（例如，Firefox、Opera、Safari 等）把 XMLHttpRequest 对象实例化为一个本地 JavaScript 对象，具体方法如下：

```
var http_request = new XMLHttpRequest();
```

为了提高程序的兼容性，可以创建一个跨浏览器的 XMLHttpRequest 对象。创建一个跨浏览器的 XMLHttpRequest 对象其实很简单，只需要判断一下不同浏览器的实现方式，如果浏览器提供了 XMLHttpRequest 类，则直接创建一个实例，否则实例化一个 ActiveX 对象，具体代码如下：

```
if (window.XMLHttpRequest) {                    // 非 IE 浏览器
   http_request = new XMLHttpRequest();
} else if (window.ActiveXObject) {              //IE 浏览器
   try {
         http_request = new ActiveXObject("Msxml2.XMLHTTP");
   } catch (e) {
         try {
               http_request = new ActiveXObject("Microsoft.XMLHTTP");
      } catch (e) {}
   }
}
```

在上面的代码中，调用 window.ActiveXObject 将返回一个对象或是 null，在 if 语句中，会把返回值看做 true 或 false（如果返回的是一个对象，则为 true，如果返回 null，则为 false）。

说明　由于 JavaScript 具有动态类型特性，而且 XMLHttpRequest 对象在不同浏览器上的实例是兼容的，所以可以用同样的方式访问 XMLHttpRequest 实例的属性的方法，不需要考虑创建该实例的方法是什么。

14.2.2　XMLHttpRequest 对象的常用方法

XMLHttpRequest 对象提供了一些常用的方法，通过这些方法可以对请求进行操作。下面对 XMLHttpRequest 对象的常用方法进行介绍。

（1）open() 方法

open() 方法用于设置进行异步请求目标的 URL、请求方法以及其他参数信息，具体语法如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
open("method","URL"[,asyncFlag[,"userName"[, "password"]]])
```

open() 方法的参数说明如表 14.1 所示。

表 14.1 open() 方法的参数说明

参 数	说 明
method	用于指定请求的类型，一般为 GET 或 POST
URL	用于指定请求地址，可以使用绝对地址或者相对地址，并且可以传递查询字符串
asyncFlag	为可选参数，用于指定请求方式，异步请求为 true，同步请求为 false，默认情况下为 true
userName	为可选参数，用于指定请求用户名，没有时可省略
password	为可选参数，用于指定请求密码，没有时可省略

例如，设置异步请求目标为 register.jsp，请求方法为 GET，请求方式为异步的代码如下：

```
http_request.open("GET","register.jsp",true);
```

（2）send() 方法

send() 方法用于向服务器发送请求。如果请求声明为异步，该方法将立即返回，否则将等到接收到响应为止。send() 方法的语法格式如下：

```
send(content)
```

- content：用于指定发送的数据，可以是 DOM 对象的实例、输入流或字符串。如果没有参数需要传递，可以设置为 null。

例如，向服务器发送一个不包含任何参数的请求，可以使用下面的代码：

```
http_request.send(null);
```

（3）setRequestHeader() 方法

setRequestHeader() 方法用于为请求的 HTTP 头设置值。setRequestHeader() 方法的具体语法格式如下：

```
setRequestHeader("header", "value")
```

- header：用于指定 HTTP 头。
- value：用于为指定的 HTTP 头设置值。

注意 *setRequestHeader() 方法必须在调用 open() 方法之后才能调用。*

例如，在发送 POST 请求时，需要设置 Content-Type 请求头的值为“application/x-www-form-urlencoded”，这时就可以通过 setRequestHeader() 方法进行设置，具体代码如下：

```
http_request.setRequestHeader("Content-Type","application/x-www-form-
urlencoded");
```

（4）abort() 方法

abort() 方法用于停止或放弃当前异步请求，其语法格式如下：

```
abort()
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（5）getResponseHeader() 方法

getResponseHeader() 方法用于以字符串形式返回指定的 HTTP 头信息，其语法格式如下：

```
getResponseHeader("headerLabel")
```

❑ headerLabel：用于指定 HTTP 头，包括 Server、Content-Type 和 Date 等。

例如，要获取 HTTP 头 Content-Type 的值，可以使用以下代码：

```
http_request.getResponseHeader("Content-Type")
```

上面的代码将获取到以下内容：

```
text/html;charset=UTF-8
```

（6）getAllResponseHeaders() 方法

getAllResponseHeaders() 方法用于以字符串形式返回完整的 HTTP 头信息，其中，包括 Server、Date、Content-Type 和 Content-Length。getAllResponseHeaders() 方法语法格式如下：

```
getAllResponseHeaders()
```

例如，应用下面的代码调用 getAllResponseHeaders() 方法，将弹出如图 14.3 所示的对话框显示完整的 HTTP 头信息。

```
alert(http_request.getAllResponseHeaders());
```

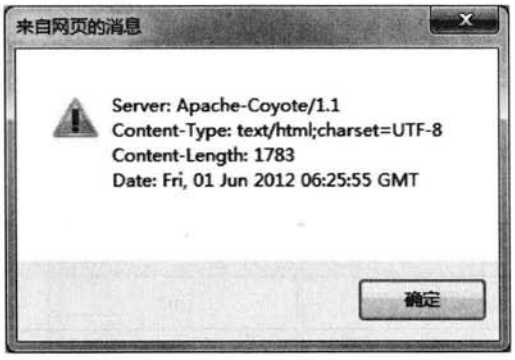


图 14.3　获取完整的 HTTP 头信息

14.2.3　XMLHttpRequest 对象的常用属性

XMLHttpRequest 对象提供了一些常用属性，通过这些属性可以获取服务器的响应状态及响应内容，下面将对 XMLHttpRequest 对象的常用属性进行介绍。

（1）onreadystatechange 属性

onreadystatechange 属性用于指定状态改变时所触发的事件处理器。在 Ajax 中，每个状态改变时都会触发这个事件处理器，通常会调用一个 JavaScript 函数。

例如，指定状态改变时触发 JavaScript 函数 getResult 的代码如下：

```
http_request.onreadystatechange = getResult;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

注意 在指定所触发的事件处理器时，所调用的 JavaScript 函数不能添加小括号及指定参数名。不过这里可以使用匿名函数。例如，要调用带参数的函数 getResult()，可以使用下面的代码：

```
http_request.onreadystatechange = function(){
   getResult("添加的参数");                    //调用带参数的函数
};                                             //通过匿名函数指定要带参数的函数
```

（2）readyState 属性

readyState 属性用于获取请求的状态。该属性共包括 5 个属性值，如表 14.2 所示。

表 14.2 readyState 属性的属性值

值	意 义	值	意 义
0	未初始化	1	正在加载
2	已加载	3	交互中
4	完成		

（3）responseText 属性

responseText 属性用于获取服务器的响应，表示为字符串。

（4）responseXML 属性

responseXML 属性用于获取服务器的响应，表示为 XML。这个对象可以解析为一个 DOM 对象。

（5）status 属性

status 属性用于返回服务器的 HTTP 状态码，常用的状态码如表 14.3 所示。

表 14.3 status 属性的状态码

值	意 义	值	意 义
200	表示成功	202	表示请求被接受，但尚未成功
400	错误的请求	404	文件未找到
500	内部服务器错误		

（6）statusText 属性

statusText 属性用于返回 HTTP 状态码对应的文本，如 OK 或 Not Fount（未找到）等。

14.3 传统Ajax的工作流程

通过前面的学习，相信大家已经对 Ajax 以及 Ajax 所使用的技术有所了解。下面将介绍在 Ajax 中如何发送请求与处理服务器响应。

14.3.1 发送请求

Ajax 可以通过 XMLHttpRequest 对象实现采用异步方式在后台发送请求。通常情况下，

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Ajax 发送请求有两种，一种是发送 GET 请求，另一种是发送 POST 请求。但是无论发送哪种请求，都需要经过以下 4 个步骤。

Step 1) 初始化 XMLHttpRequest 对象。为了提高程序的兼容性，需要创建一个跨浏览器的 XMLHttpRequest 对象，并且判断 XMLHttpRequest 对象的实例是否成功，如果不成功，则给予提示，具体代码如下：

```
http_request = false;
if (window.XMLHttpRequest) {                                    // 非 IE 浏览器
      http_request = new XMLHttpRequest();                      // 创建 XMLHttpRequest 对象
} else if (window.ActiveXObject) {                              // IE 浏览器
      try {
          http_request = new ActiveXObject("Msxml2.XMLHTTP");
                                                                // 创建 XMLHttpRequest 对象
      } catch (e) {
          try {
              // 创建 XMLHttpRequest 对象
              http_request = new ActiveXObject("Microsoft.XMLHTTP");
          } catch (e) {}
      }
  }
if (!http_request) {
      alert(" 不能创建 XMLHttpRequest 对象实例！ ");
      return false;
}
```

Step 2) 为 XMLHttpRequest 对象指定一个返回结果处理函数（即回调函数），用于对返回结果进行处理，具体代码如下：

```
http_request.onreadystatechange = getResult;                    // 调用返回结果处理函数
```

Step 3) 创建一个与服务器的连接。在创建时，需要指定发送请求的方式（即 GET 或 POST），以及设置是否采用异步方式发送请求。

例如，采用异步方式发送 GET 方式的请求的具体代码如下：

```
http_request.open('GET', url, true);
```

例如，采用异步方式发送 POST 方式的请求的具体代码如下：

```
http_request.open('POST', url, true);
```

说明 在 open() 方法中的 url 参数，可以是一个 JSP 页面的 URL 地址，也可以是 Servlet 的映射地址。也就是说，请求处理页，可以是一个 JSP 页面，也可以是一个 Servlet。

说明 在指定 URL 参数时，最好将一个时间戳追加到该 URL 参数的后面，这样可以防止因浏览器缓存结果而不能实时得到最新的结果。例如，可以指定 URL 参数为以下代码：

```
String url="deal.jsp?nocache="+new Date().getTime();
```

Step 4) 向服务器发送请求。XMLHttpRequest 对象的 send() 方法可以实现向服务器发送请求，该方法需要传递一个参数，如果发送的是 GET 请求，可以将该参数设置为 null，如果发送的是 POST 请求，可以通过该参数指定要发送的请求参数。

向服务器发送 GET 请求的代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
http_request.send(null);                                    // 向服务器发送请求
```

向服务器发送 POST 请求的代码如下：

```
var param="user="+form1.user.value+"&pwd="+form1.pwd.value+"&email="+form1.email.
value;                                                      // 组合参数
http_request.send(param);                                   // 向服务器发送请求
```

需要注意的是，在发送 POST 请求前，还需要设置正确的请求头，具体代码如下：

```
http_request.setRequestHeader("Content-Type","application/x-www-form-
urlencoded");
```

上面的这句代码，需要添加在 http_request.send(param); 语句之前。

14.3.2 处理服务器响应

当向服务器发送请求后，接下来就需要处理服务器响应了。在向服务器发送请求时，需要通过 XMLHttpRequest 对象的 onreadystatechange 属性指定一个回调函数，用于处理服务器响应。在这个回调函数中，首先需要判断服务器的请求状态，保证请求已完成，然后再根据服务器的 HTTP 状态码，判断服务器对请求的响应是否成功，如果成功，则获取服务器的响应反馈给客户端。

XMLHttpRequest 对象提供了两个用来访问服务器响应的属性，一个是 responseText 属性，用于返回字符串响应，另一个是 responseXML 属性，用于返回 XML 响应。

1. 处理字符串响应

字符串响应通常应用在响应不是特别复杂的情况下。例如，将响应显示在提示对话框中，或者响应只是显示成功或失败的字符串。

将字符串响应显示到提示对话框中的回调函数的具体代码如下：

```
function getResult() {
    if (http_request.readyState == 4) {                     // 判断请求状态
        if (http_request.status == 200) {                   // 请求成功，开始处理返回结果
            alert(http_request.responseText);               // 显示判断结果
        } else {                                            // 请求页面有错误
            alert("您所请求的页面有错误！");
        }
    }
}
```

如果需要将响应结果显示到页面的指定位置，也可以先在页面的合适位置添加一个 或 标记，设置该标记的 id 属性，例如 div_result，然后在回调函数中应用以下代码显示响应结果。

```
document.getElementById("div_result").innerHTML=http_request.responseText;
```

2. 处理 XML 响应

如果在服务器端需要生成特别复杂的响应，那么就需要应用 XML 响应。应用 XMLHttpRequest 对象的 responseXML 属性，可以生成一个 XML 文档，而且当前浏览器已经提供了很好的解析 XML 文档对象的方法。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

例如，有一个保存图书信息的 XML 文档，具体代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
   <books>
          <book>
                 <title>Java Web 开发典型模块大全 </title>
                 <publisher> 人民邮电出版社 </publisher>
          </book>
          <book>
                 <title>Java 范例完全自学手册 </title>
                 <publisher> 人民邮电出版社 </publisher>
          </book>
   </books>
```

在回调函数中遍历保存图书信息的 XML 文档，并显示到页面中的代码如下：

```
function getResult() {
    if (http_request.readyState == 4) {                    // 判断请求状态
        if  (http_request.status == 200) {                 // 请求成功，开始处理响应
            var xmldoc = http_request.responseXML;
            var str="";
            for(i=0;i<xmldoc.getElementsByTagName("book").length;i++){
                var book = xmldoc.getElementsByTagName("book").item(i);
                str=str+"《"+book.getElementsByTagName("title")[0].
                firstChild.data+"》由 ""+book.getElementsByTagName('publisher')
                [0].firstChild.data+"" 出版 <br>";
                }
                document.getElementById("book").innerHTML=str;
                                                           // 显示图书信息
        } else {                                           // 请求页面有错误
            alert(" 您所请求的页面有错误！ ");
        }
    }
}
<div id="book"></div>
```

通过上面的代码获取的 XML 文档的信息如下：

```
《Java Web 开发典型模块大全》由“人民邮电出版社”出版
《Java 范例完全自学手册》由“人民邮电出版社”出版
```

14.3.3　一个完整的实例——检测用户名是否唯一

【例 14.1】 编写一个会员注册页面，并应用 Ajax 实现检测用户名是否唯一的功能。

实例位置：光盘 \MR\ 源码 \ 第 14 章 \14.1

Step 1) 创建 index.jsp 文件，在该文件中添加一个用于收集用户注册信息的表单及表单元素，以及代表“检测用户名”按钮的图片，并在该图片的 onClick 事件中调用 checkName() 方法，检测用户名是否被注册，关键代码如下：

```
<form method="post" action="" name="form1">
用户名：<input name="username" type="text" id="username" size="32">
<img src="images/checkBt.jpg" width="104" height="23" style="cursor:pointer;"
 onClick="checkUser(form1.username);">
密码：<input name="pwd1" type="password" id="pwd1" size="35"><
确认密码：<input name="pwd2" type="password" id="pwd2" size="35">
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
E-mail: <input name="email" type="text" id="email" size="45">
<input type="image" name="imageField" src="images/registerBt.jpg">
</form>
```

Step 2 在页面的合适位置添加一个用于显示提示信息的 标记，并且通过 CSS 设置该 标记的样式，关键代码如下：

```
<div id="toolTip"></div>
```

Step 3 编写一个自定义的 JavaScript 函数 createRequest()，在该函数中，首先初始化 XMLHttpRequest 对象，然后指定处理函数，再创建与服务器的连接，最后向服务器发送请求。createRequest() 函数的具体代码如下：

```
function createRequest(url) {
    http_request = false;
    if (window.XMLHttpRequest) {                          // 非 IE 浏览器
        http_request = new XMLHttpRequest();              // 创建 XMLHttpRequest 对象
    } else if (window.ActiveXObject) {                    // IE 浏览器
        try {
          http_request = new ActiveXObject("Msxml2.XMLHTTP");
                                                          // 创建 XMLHttpRequest 对象
        } catch (e) {
            try {
                http_request = new ActiveXObject("Microsoft.XMLHTTP");
                                                          // 创建 XMLHttpRequest 对象
            } catch (e) {}
        }
    }
    if (!http_request) {
        alert(" 不能创建 XMLHttpRequest 对象实例！ ");
        return false;
    }
    http_request.onreadystatechange = getResult;          // 调用返回结果处理函数
    http_request.open('GET', url, true);                  // 创建与服务器的连接
    http_request.send(null);                              // 向服务器发送请求
}
```

Step 4 编写回调函数 getResult()，该函数主要根据请求状态对返回结果进行处理。在该函数中，如果请求成功，为提示框设置相应的提示内容，并且让该提示框显示。getResult() 函数的具体代码如下：

```
function getResult() {
    if (http_request.readyState == 4) {                   // 判断请求状态
        if (http_request.status == 200) {                 // 请求成功，开始处理返回结果
          document.getElementById("toolTip").innerHTML=http_request.responseText;
                                                          // 设置提示内容
            document.getElementById("toolTip").style.display="block";
                                                          // 显示提示框
        } else {                                          // 请求页面有错误
            alert(" 您所请求的页面有错误！ ");
        }
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Step 5) 编写自定义的 JavaScript 函数 checkUser()，用于检测用户名是否为空，当用户名不为空时，调用 createRequest() 函数发送异步请求检测用户名是否被注册。checkUser() 函数的具体代码如下：

```
function checkUser(userName){
if(userName.value==""){
   alert("请输入用户名！ ");userName.focus();return;
}else{
   createRequest('checkUser.jsp?user='+ encodeURIComponent(userName.value));
   }
}
```

Step 6) 编写检测用户名是否被注册的处理页 checkUser.jsp，在该页面中判断输入的用户名是否注册，并应用 JSP 内置对象 out 的 println() 方法输出判断结果。checkUser.jsp 页面的具体代码如下：

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8" %>
<%
   String[] userList={"明日科技","mr","mrsoft","wgh"};          //创建一个一维数组
   //获取用户名
   String user=new String(request.getParameter("user").
   getBytes("ISO-8859-1"),"UTF-8");
   Arrays.sort(userList);                                       //对数组排序
   int result=Arrays.binarySearch(userList,user);               //搜索数组
   if(result>-1){
          out.println("很抱歉，该用户名已经被注册！ ");          //输出检测结果
   }else{
          out.println("恭喜您，该用户名没有被注册！ ");          //输出检测结果
   }
%>
```

运行本实例，在用户名文本框中输入 mr，单击“检查用户名”按钮，将显示如图 14.4 所示的提示信息。



图 14.4　用户名被注册时显示的效果

14.4　利用jQuery实现Ajax

通过前面的介绍，我们可以知道在 Web 中应用 Ajax 的工作流程比较烦琐，每次都需要编

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

写大量的 JavaScript 代码。不过应用目前比较流行的 jQuery 可以简化 Ajax。下面将具体介绍如何应用 jQuery 实现 Ajax。

14.4.1 jQuery 简介

jQuery 是一套简洁、快速、灵活的 JavaScript 脚本库，它是由 John Resig 于 2006 年创建的，它帮助我们简化了 JavaScript 代码。JavaScript 脚本库类似于 Java 的类库，我们将一些工具方法或对象方法封装在类库中，方便用户使用。jQuery 因为它的简便易用，已被大量的开发人员推崇。

要在自己的网站中应用 jQuery 库，需要下载并配置它。

1. 下载和配置 jQuery

jQuery 是一个开源的脚本库，我们可以在它的官方网站（http://jquery.com）中下载到最新版本的 jQuery 库。当前的版本是 1.7.2，下载后将得到名称为 jquery-1.7.2.min.js 的文件。

将 jQuery 库下载到本地计算机后，还需要在项目中配置 jQuery 库。将下载后的 jquery-1.7.2.min.js 文件放置到项目的指定文件夹中，通常放置在 JS 文件夹中，然后在需要应用 jQuery 的页面中使用下面的语句，将其引用到文件中。

```
<script language="javascript" src="JS/jquery-1.7.2.min.js"></script>
或者
<script src="JS/jquery-1.7.2.min.js" type="text/javascript"></script>
```

注意 引用 jQuery 的 <script> 标签，必须放在所有的自定义脚本文件的 <script> 之前，否则在自定义的脚本代码中应用不到 jQuery 脚本库。

2. jQuery 的工厂函数

在 jQuery 中，无论我们使用哪种类型的选择符，都需要从一个“$”符号和一对“()”开始。在“()”中通常使用字符串参数，参数中可以包含任何 CSS 选择符表达式。下面介绍几种比较常见的用法。

❑ 在参数中使用标记名。

$("div")：用于获取文档中全部的 。

❑ 在参数中使用 ID。

$("#username")：用于获取文档中 ID 属性值为 username 的一个元素。

❑ 在参数中使用 CSS 类名。

$(".btn_grey")：用于获取文档中使用 CSS 类名为 btn_grey 的所有元素。

3. 我的第一个 jQuery 脚本

【例 14.2】 应用 jQuery 弹出一个提示对话框。

实例位置：光盘 \MR\ 源码 \ 第 14 章 \14.2

Step 1) 在 Eclipse 中创建动态 Web 项目，并在该项目的 WebContent 节点下创建一个名称为 JS 的文件夹，将 jquery-1.7.2.min.js 复制到该文件夹中。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明　默认情况下，在 Eclipse 创建的动态 Web 项目中，添加 jQuery 库以后，将出现红 ×，标识有语法错误，但是程序仍然可以正常运行。解决该问题的方法是：首先在 Eclipse 的主菜单中选择“窗口”/“首选项目”菜单项，将打开“首选项”对话框，并在“首选项”对话框的左侧选择 JavaScript/“验证器”/“错误 / 警告”节点，然后将右侧的 Enable JavaScript Semantic Validation 复选框取消选取状态，并应用，接下来再找到项目的 .project 文件，将其中的以下代码删除：

```
<buildCommand>
<name>org.eclipse.wst.jsdt.core.javascriptValidator</name>
<arguments>
</arguments>
</buildCommand>
```

并保存该文件，最后重新添加 jQuery 库就可以了。

Step 2　创建一个名称为 index.jsp 的文件，在该文件的 <head> 标记中引用 jQuery 库文件，关键代码如下：

```
<script type="text/javascript" src="JS/jquery-1.7.2.min.js"></script>
```

Step 3　在 <body> 标记中，应用 HTML 的 <a> 标记添加一个空的超链接，关键代码如下：

```
<a href="#"> 弹出提示对话框 </a>
```

Step 4　编写 jQuery 代码，实现在单击页面中的超链接时，弹出一个提示对话框，具体代码如下：

```
<script>
$(document).ready(function(){
    // 获取超链接对象，并为其添加单击事件
    $("a").click(function(){
            alert(" 我的第一个 jQuery 脚本！ ");
    });
});
</script>
```

实际上，上面的代码还可以更简单，也就是将 $(document).ready 用 "$" 符代替，替换后的代码如下：

```
<script>
$(function(){
    // 获取超链接对象，并为其添加单击事件
    $("a").click(function(){
            alert(" 我的第一个 jQuery 脚本！ ");
    });
});
</script>
```

Step 5　运行本实例，单击页面中的“弹出提示对话框”超链接，将弹出如图 14.5 所示的提示对话框。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

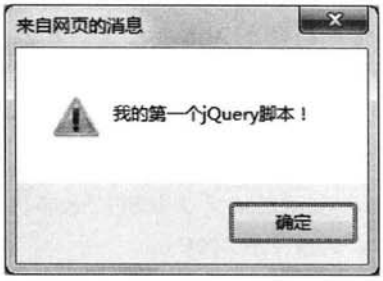


图 14.5　弹出的提示对话框

在第 2 章中，介绍 JavaScript 的事件时，我们知道，要实现例 14.2 的效果，还可以通过下面的代码实现：

```
<script>
window.onload=function(){
   $("a").click(function(){
          alert("我的第一个jQuery脚本！");
   });
}
</script>
```

这时，读者可能要问，这两种方法有什么区别，究竟哪种方法更好呢？下面我们就来介绍二者的区别。window.onload() 方法是在页面所有的东西都载入完毕后才会执行，例如，图片和横幅等。而 $(document).ready() 方法则是在 DOM 元素载入就绪后执行。在一个页面中可以放置多个 $(document).ready() 方法，而 window.onload() 方法在页面上只允许放置一个（常规情况）。这两个方法可以同时在页面中执行，两者并不矛盾。不过，$(document).ready() 方法比 window.onload() 方法载入更快。

14.4.2　应用 load() 方法发送请求

load() 方法通过 AJAX 请求从服务器加载数据，并把返回的数据放置到指定的元素中，它的语法格式如下：

```
.load( url [, data] [, complete(responseText, textStatus, XMLHttpRequest)] )
```

- url：用于指定要请求页面的 URL 地址。
- data：可选参数，用于指定跟随请求一同发送的数据。因为 load() 方法不仅可以导入静态的 HTML 文件，还可以导入动态脚本（例如 JSP 文件），当要导入动态文件时，就可以把要传递的数据通过该参数进行指定。
- complete(responseText, textStatus, XMLHttpRequest)：用于指定调用 load() 方法并得到服务器响应后，再执行的另外一个函数。如果不指定该参数，那么服务器响应完成后，则直接将匹配元素的 HTML 内容设置为返回的数据。该函数的三个参数中，responseText 表示请求返回的内容，textStatus 表示请求状态，XMLHttpRequest 表示 XMLHttpRequest 对象。

例如，要请求名称为 book.html 的静态页面，可以使用下面的代码：

```
$("#getBook").load("book.html");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明 使用 load() 方法发送请求时，有两种方式，一种是 GET 请求，另一种是 POST 请求。采用哪种请求方式，将由 data 参数的值决定。当 load() 方法没有向服务器传递参数时，请求的方式就是 GET，反之请求的方式就是 POST。

【例 14.3】 应用 jQuery 在页面中显示实时走动的时间。

实例位置：光盘 \MR\ 源码 \ 第 14 章 \14.3

Step 1) 在 Eclipse 中创建动态 Web 项目，并在该项目的 WebContent 节点下创建一个名称为 JS 的文件夹，将 jquery-1.7.2.min.js 复制到该文件夹中。

Step 2) 创建一个名称为 index.jsp 的文件，在该文件的 <head> 标记中引用 jQuery 库文件，关键代码如下：

```
<script type="text/javascript" src="JS/jquery-1.7.2.min.js"></script>
```

Step 3) 在 <body> 标记中，添加一个 id 为 getTime 的 标记，关键代码如下：

```
<div id="getTime"> 正在获取时间 ...</div>
```

Step 4) 编写 jQuery 代码，通过每隔一秒钟请求一次的 getTime.jsp 文件获取当前系统时间，具体代码如下：

```
<script>
$(document).ready(function(){
          window.setInterval("$('#getTime').load('getTime.jsp',{});",1000);
});
</script>
```

Step 5) 创建一个名称为 getTime.jsp 的文件，在该文件中，编写用于在页面中输出当前系统时间的 JSP 代码。getTime.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"pageEncoding="UTF-8"%>
<%@page import="java.util.Date"%>
<%
   out.println(new java.text.SimpleDateFormat("YYYY-MM-dd HH:mm:ss").format(new
   Date()));                                                    // 输出系统时间
%>
```

运行本实例，在页面中将显示如图 14.6 所示的走动的当前时间。

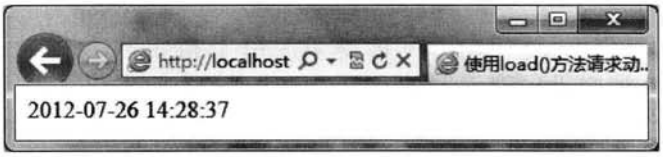


图 14.6　显示走动的当前时间

使用 load() 方法请求 HTML 页面时，也可以只加载被请求文档中的一部分。这可以通过在请求的 URL 地址后面接一个空格，再加上要加载内容的 jQuery 选择器。例如，要载 book.html 页面中 ID 属性为 javaweb 的元素的内容，可以使用下面的代码。

```
$("#getTime").load("book.html #javaweb");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

如果在 book.html 文件的添加以下代码：

```
<ul id="javaweb">
   <li>Java Web 开发实战宝典 </li>
   <li>Java Web 典型模块与项目实战 </li>
</ul>
<ul id="java">
   <li>Java 从入门到精通 </li>
   <li>Java 典型模块精解 </li>
</ul>
```

在运行代码 $("#getTime").load("book.html #javaweb"); 时，将显示如图 14.7 所示的运行结果。



图 14.7　加载请求文档中的 id 属性为 javaweb 的 标记

14.4.3　发送 GET 和 POST 请求

在 jQuery 中，虽然提供了 load() 方法可以根据提供的参数发送 GET 和 POST 请求，但是该方法有一定的局限性，它是一个局部方法，需要在 jQuery 包装集上调用，并且会将返回的 HTML 加载到对象中，即使设置了回调函数也还是会加载。为此，jQuery 还提供了全局的、专门用于发送 GET 请求和 POST 请求的 get() 方法和 post() 方法。

（1）get() 方法

$.get() 方法用于通过 GET 方式来进行异步请求，它的语法格式如下：

```
$.get(url [, data] [, success(data, textStatus, jqXHR)] [, dataType] )
```

- url：字符串类型的参数，用于指定请求页面的 URL 地址。
- data：可选参数，用于指定发送至服务器的key/value数据。data参数会自动添加到url中。如果 url 中的某个参数又通过 data 参数进行传递，那么 get() 方法是不会自动合并相同名称的参数的。
- success(data,textStatus,jqXHR)：可选参数，用于指定载入成功后执行的回调函数。其中，data 用于保存返回的数据，testStatus 为状态码（可以是 timeout，error，notmodified，success 或 parsererror），jqXHR 为 XMLHTTPRequest 对象。不过该回调函数只有当 testStatus 的值为 success 时才会执行。
- dataType：可选参数，用于指定返回数据的类型，可以是 XML、JSON、SCRIPT 或者 HTML，默认值为 HTML。

例如，使用 get() 方法请求 deal.jsp，并传递两个字符串类型的参数，可以使用下面的代码：

```
$.get("deal.jsp",{name:" 无语 ",branch:"java"});
```

【例 14.4】 将例 14.1 的程序修改为采用 jQuery 的 get() 方法发送请求的方式来实现。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

实例位置：光盘\MR\源码\第14章\14.4

Step 1) 在 Eclipse 中创建动态 Web 项目，并在该项目的 WebContent 节点下创建一个名称为 JS 的文件夹，将 jquery-1.7.2.min.js 复制到该文件夹中。

Step 2) 创建一个名称为 index.jsp 的文件，在该文件的 <head> 标记中引用 jQuery 库文件，关键代码如下：

```
<script type="text/javascript" src="JS/jquery-1.7.2.min.js"></script>
```

Step 3) 在 <body> 标记中，添加一个用于收集用户注册信息的表单及表单元素，以及代表“检测用户名”按钮的图片，并为这个图片设置 id 属性，关键代码如下：

```
<form method="post" action="" name="form1">
用户名：<input name="username" type="text" id="username" size="32">
<img id="checkuser" src="images/checkBt.jpg"
          width="104" height="23" style="cursor: pointer;">
密码：<input name="pwd1" type="password" id="pwd1" size="35"><
确认密码：<input name="pwd2" type="password" id="pwd2" size="35">
E-mail：<input name="email" type="text" id="email" size="45">
<input type="image" name="imageField" src="images/registerBt.jpg">
</form>
```

Step 4) 在页面的合适位置添加一个用于显示提示信息的 标记，并且通过 CSS 设置该 标记的样式。由于此处的代码与例 14.1 完全相同，所以这里不再给出。

Step 5) 在引用 jQuery 库的代码下方，编写 JavaScript 代码，实现当 DOM 元素载入就绪后，为代表“检测用户名”的按钮图片添加单击事件，在该单击事件中，判断用户名是否为空，如果为空，则给出提示对话框，并让该文本框获得焦点，否则应用 get() 方法，发送异步请求检测用户名是否被注册，具体代码如下：

```
<script type="text/javascript">
   $(document).ready(function(){
          $("#checkuser").click(function(){
                 if ($("#username").val()== "") {// 判断是否输入用户名
                        alert("请输入用户名！");
                        $("#username").focus();// 让用户名文本框获得焦点
                        return;
                 } else {                          // 已经输入用户名时，检测用户名是否唯一
                        $.get("checkUser.jsp",
                                      {user:$("#username").val()},
                                      function(data){
                                             $("#toolTip").text(data);
                                                    // 设置提示内容
                                             $("#toolTip").show();
                                                    // 显示提示框
                        });
                 }
          });
   });
</script>
```

Step 6) 编写检测用户名是否被注册的处理页 checkUser.jsp，在该页面中判断输入的用户名是否注册，并应用 JSP 内置对象 out 的 println() 方法输出判断结果。由于此处的代码与

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

14.1 完全相同，这里不再给出。

运行本实例，在用户名文本框中输入 mr，单击“检测用户名”按钮，将显示如图 14.4 所示的提示信息。

从这个程序中，我们可以看到使用 jQuery 替代传统的 Ajax，确实简单、方便了许多。它可使开发人员的精力不必集中于实现 Ajax 功能的烦琐步骤，而专注于程序的功能。

说明 get() 方法通常用来实现简单的 GET 请求功能，对于复杂的 GET 请求需要使用 $.ajax() 方法实现。例如，在 get() 方法中指定的回调函数，只能在请求成功时调用，如果需要在出错时也要执行一个函数，那么就需要使用 $.ajax() 方法实现。$.ajax() 方法将在 14.4.5 节进行介绍。

（2）post() 方法

$.post() 方法用于通过 POST 方式进行异步请求，它的语法格式如下：

```
$.post( url [, data] [, success(data, textStatus, jqXHR)] [, dataType] )
```

- url：字符串类型的参数，用于指定请求页面的 URL 地址。
- data：可选参数，用于指定发送到服务器的 key/value 数据，该数据将连同请求一同被发送到服务器。
- success(data, textStatus, jqXHR)：可选参数，用于指定载入成功后执行的回调函数。在回调函数中含有两个参数，分别是 data（返回的数据）和 testStatus（状态码，可以是 timeout，error，notmodified，success 或 parsererror）。不过该回调函数只有当 testStatus 的值为 success 时才会执行。
- dataType：可选参数，用于指定返回数据的类型，可以是 XML、JSON、SCRIPT、TEXT 或 HTML，默认值为 HTML。

例如，使用 post() 方法请求 deal.jsp，并传递两个字符串类型的参数和回调函数，可以使用下面的代码：

```
$.post("deal.jsp",{title:"祝福",content:"祝愿天下的所有母亲平安、健康…
"},function(data){
   alert(data);
});
```

【例 14.5】 实现实时显示聊天内容。

实例位置：光盘 \MR\ 源码 \ 第 14 章 \14.5

Step 1 在 Eclipse 中创建动态 Web 项目，并在该项目的 WebContent 节点下创建一个名称为 JS 的文件夹，将 jquery-1.7.2.min.js 复制到该文件夹中。

Step 2 创建一个名称为 index.jsp 的文件，在该文件的 <head> 标记中引用 jQuery 库文件，关键代码如下：

```
<script type="text/javascript" src="JS/jquery-1.7.2.min.js"></script>
```

Step 3 在 index.jsp 页面的合适位置添加一个 标记，用于显示聊天内容，具体代码如下：

```
<div id="content" style="height:206px; overflow:hidden;"> 欢迎光临碧海聆音聊天室！ </div>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Step 4 在引用 jQuery 库的代码下方，编写一个名称为 getContent() 的自定义 JavaScript 函数，用于发送 GET 请求读取聊天内容并显示。getContent() 函数的具体代码如下：

```
function getContent() {
    $.get("ChatServlet?action=get&nocache=" + new Date().getTime(),
                    function(data) {
                            $("#content").html(data);        //显示读取到的聊天内容
                    });
}
```

Step 5 创建并配置一个聊天信息相关的 Servlet 实现类 ChatServlet，并在该 Servlet 中编写 get() 方法获取全部聊天信息。get() 方法的具体代码如下：

```
public void get(HttpServletRequest request,HttpServletResponse response) throws
ServletException,IOException{
    response.setContentType("text/html;charset=UTF-8");  //设置响应的内容类型
                                                           及编码方式
    response.setHeader("Cache-Control", "no-cache");     //禁止页面缓存
    PrintWriter out = response.getWriter();              //获取输出流对象
    /********************获取聊天信息*****************************/
    ServletContext application=getServletContext();      //获取 application 对象
    String msg="";
    if(null!=application.getAttribute("message")){
            Vector<String> v_temp=(Vector<String>)application.getAttribute
            ("message");
            for(int i=v_temp.size()-1;i>=0;i--){
                    msg=msg+"<br>"+v_temp.get(i);
            }
    }else{
            msg="欢迎光临碧海聆音聊天室！";
    }
    out.println(msg);                                    //输出生成后的聊天信息
    out.close();                                         //关闭输出流对象
}
```

Step 6 为了实现实时显示最新的聊天内容，当 DOM 元素载入就绪后，需要在 index.jsp 文件的引用 jQuery 库的代码下方编写下面的代码。

```
$(document).ready(function() {
    getContent();                                        //获取聊天内容
    window.setInterval("getContent();", 5000);           //每隔 5 秒钟获取一次聊天内容
});
```

Step 7 在 index.jsp 页面的合适位置添加用于获取用户昵称和说话内容的表单及表单元素，关键代码如下：

```
<form name="form1" method="post" action="">
    <input name="user" type="text" id="user" size="20"> 说:
<input name="speak" type="text" id="speak" size="50">
      <input id="send" type="button" class="btn_grey" value="发送">
</form>
```

Step 8 在引用 jQuery 库的代码下方，编写 JavaScript 代码，实现当 DOM 元素载入就绪后，

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

为“发送”按钮添加单击事件，在该单击事件中，判断昵称和发送信息文本框是否为空，如果为空，则给出提示对话框，并让该文本框获得焦点，否则应用 post() 方法，发送异步请求到服务器，保存聊天信息。具体代码如下：

```
$(document).ready(function() {
        $("#send").click(function() {
                if ($("#user").val() == "") {          //判断昵称是否为空
                        alert("请输入您的昵称！");
                }
                if ($("#speak").val() == "") {         //判断说话内容是否为空
                        alert("说话内容不可以为空！");
                        $("speak").focus();            //让说话内容文本框获得焦点
                }
                $.post("ChatServlet?action=send", {
                        user : $("#user").val(),
                        speak : $("#speak").val()
                });                                    //发送 POST 请求
                $("#speak").val("");                   //清空说话内容文本框的值
                $("#speak").focus();                   //让说话内容文本框获得焦点
        });
});
```

Step 9 在聊天信息相关的 Servlet 实现类 ChatServlet 中，编写 send() 方法将聊天信息保存到 application 中。send() 方法的具体代码如下：

```
public void send(HttpServletRequest request,HttpServletResponse response)
 throws ServletException, IOException {
   ServletContext application=getServletContext();
                                                   //获取 application 对象
   /********************保存聊天信息*****************************/
   response.setContentType("text/html;charset=UTF-8");
   String user=request.getParameter("user");      //获取用户昵称
   String speak=request.getParameter("speak");    //获取说话内容
   Vector<String> v=null;
   String message="["+user+"]说："+speak;          //组合说话内容
   if(null==application.getAttribute("message")){
          v=new Vector<String>();
   }else{
          v=(Vector<String>)application.getAttribute("message");
   }
   v.add(message);
   application.setAttribute("message", v);        //将聊天内容保存到 application 中
Random random = new Random();
request.getRequestDispatcher("ChatServlet?action=get&nocache="+ random.
nextInt(10000)).forward(request, response);
}
```

运行本实例，在页面中将显示最新的聊天内容，如图 14.8 所示。如果当前聊天室内没有任何聊天内容，将显示“欢迎光临碧海聆音聊天室！”。当用户输入昵称及说话内容后，单击“发送”按钮，将发送聊天内容，并显示到上方的聊天内容列表中。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

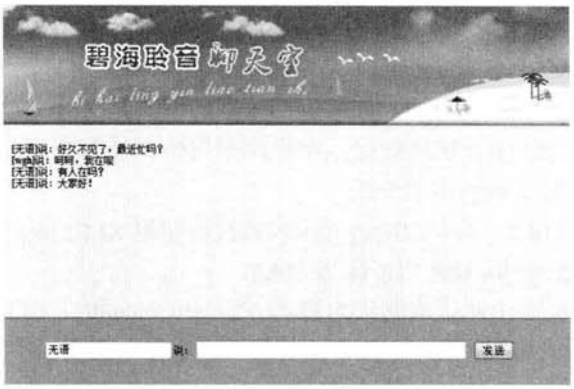


图 14.8　实时显示聊天内容

14.4.4　服务器返回的数据格式

服务器端处理完客户端的请求后，会为客户端返回一个数据，这个返回数据的格式可以是很多种，在 $.get() 方法和 $.post() 方法中就可以设置服务器返回数据的格式。常用的格式有 HTML、XML、JSON 这三种格式。

1．HTML 片段

如果返回的数据格式为 HTML 片段，在回调函数中数据不需要进行任何处理就可以直接使用，而且在服务器端也不需要做过多的处理。例如，在例 14.5 中，读取聊天信息时，我们使用的是 get() 方法与服务器进行交互，并在回调函数中处理返回数据类型为 HTML 的数据，关键代码如下：

```
$.get("ChatServlet?action=get&nocache=" + new Date().getTime(),
        function(date) {
                $("#content").html(date);                    //显示读取到的聊天内容
}
);
```

在上面的代码中，并没有使用 get() 方法的第 4 个参数 dataType 来设置返回数据的类型，因为数据类型默认就是 HTML 片段。

如果返回数据的格式为 HTML 片段，那么返回数据 data 不需要进行任何的处理，直接应用在 html() 方法中即可。在 Servlet 中也不必对处理后的数据进行任何加工，只需要设置响应的内容类型为 text/html 即可。例如，例 14.5 中获取聊天信息的 Servlet 代码，这里我们只是设置了响应的内容类型，以及将聊天内容输出到响应中。

```
response.setContentType("text/html;charset=UTF-8");
response.setHeader("Cache-Control", "no-cache");//禁止页面缓存
PrintWriter out = response.getWriter();
String msg=" 欢迎光临碧海聆音聊天室！ ";                    // 这里定义一个变量模拟生成的聊天信息
out.println(msg);
out.close();
```

使用 HTML 片段作为返回数据类型，实现起来比较简单，但是它有一个致命的缺点，那就是这种数据结构方式不一定能在其他的 Web 程序中得到重用。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2．XML 数据

XML（Extensible Markup Language）是一种可扩展的标记语言，它强大的可移植性和可重用性都是其他的语言所无法比拟的。如果返回数据的格式是 XML 文件，那么在回调函数中就需要对 XML 文件进行处理和解析数据。在程序开发时，经常应用 attr() 方法获取节点的属性；find() 方法获取 XML 文档的文本节点。

【例 14.6】 修改例 14.5，将获取的聊天内容修改为使用 XML 格式返回数据。

实例位置：光盘\MR\源码\第 14 章\14.6

Step 1) 修改 index.jsp 页面中的读取聊天内容的方法 getContent()，设置 get() 方法的返回数据的格式为 XML，将返回的 XML 格式的聊天内容显示到页面中。修改后的代码如下：

```
function getContent() {
   $.get("ChatServlet?action=get&nocache=" + new Date().getTime(),
                  function(data) {
                         var msg="";                          //初始化聊天内容字符串
                         $(data).find("message").each(function(){
                                msg+="<br>"+$(this).text();
                                                              //读取一条留言信息
                         });
                  $("#content").html(msg);                    //显示读取到的聊天内容
                  },"XML");
}
```

Step 2) 修改 ChatServlet 中获取全部聊天信息的 get() 方法，将聊天内容以 XML 格式输出，修改后的代码如下：

```
public void get(HttpServletRequest request,HttpServletResponse response) throws
       ServletException,IOException{
response.setContentType("text/xml;charset=UTF-8");       //设置响应的内容类型及编码方式
PrintWriter out = response.getWriter();          //获取输出流对象
out.println("<?xml version='1.0'?>");
out.println("<chat>");
/********************获取聊天信息*****************************/
ServletContext application=getServletContext(); //获取 application 对象
if(null!=application.getAttribute("message")){
      Vector<String> v_temp=(Vector<String>)application.getAttribute("message");
           for(int i=v_temp.size()-1;i>=0;i--){
                  out.println("<message>"+v_temp.get(i)+"</message>");
           }
   }else{
           out.println("<message>欢迎光临碧海聆音聊天室！</message>");
   }
   out.println("</chat>");
   out.flush();
   out.close();                                               //关闭输出流对象
}
```

运行本实例，同样可以得到如图 14.8 所示的运行结果。

虽然 XML 的可重用性和可移植性比较强，但是 XML 文档的体积较大，与其他格式的文档相比，解析和操作 XML 文档要相对慢一些。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

3．JSON 数据

JSON（JavaScript Object Notation）是一种轻量级的数据交换格式。语法简洁，不仅易于阅读和编写，而且也易于机器的解析和生成。读取 JSON 文件的速度也非常快。正是由于 XML 文档的体积过于庞大和它较为复杂的操作性，才诞生了 JSON。与 XML 文档一样，JSON 文件也具有很强的重用性，而且相对于 XML 文件而言，JSON 文件的操作更加方便、体积更为小巧。

JSON 由两个数据结构组成，一种是对象（“名称 / 值”形式的映射），另一种是数组（值的有序列表）。JSON 没有变量或其他控制，只用于数据传输。

（1）对象

在 JSON 中，可以使用下面的语法格式来定义对象。

```
{"属性 1": 属性值 1,"属性 2": 属性值 2…, "属性 n": 属性值 n}
```

- 属性 1 ～属性 n：用于指定对象拥有的属性名。
- 属性值 1 ～属性值 n：用于指定各属性对应的属性值，其值可以是字符串、数字、布尔值（true/false）、null、对象和数组。

例如，定义一个保存人员信息的对象，可以使用下面的代码：

```
{
"name":"wgh",
"email":"wgh717@sohu.com",
"address":"长春市"
}
```

（2）数组

在 JSON 中，可以使用下面的语法格式来定义对象。

```
{"数组名":[
对象 1, 对象 2,…, 对象 n
]}
```

- 数组名：用于指定当前数组名。
- 对象 1 ～对象 n：用于指定各数组元素，它的值为合法的 JSON 对象。

例如，定义一个保存会员信息的数组，可以使用下面的代码：

```
{"member":[
   {"name":"wgh","address":"长春市","email":"wgh717@sohu.com"},
{"name":"明日科技","address":"长春市","email":"mingrisoft@mingrisoft.com"}
]}
```

这段 JSON 数据在 XML 中的表现形式为：

```
<?xml version="1.0" encoding="UTF-8"?>
<people>
   <name>明日科技</name>
   <address>长春市</branch>
   <email>mingrisoft@mingrisoft.com</email>
</people>
<people>
   <name>wgh</name>
   <address>长春市</branch>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    <email>wgh717@sohu.com</email>
</people>
```

在数据量较大的时候，就可以看出 JSON 数据格式相对于 XML 格式的优势，而且 JSON 数据格式的结构更加清晰。

【例 14.7】 修改例 14.5，将获取的聊天内容修改为使用 JSON 格式返回数据。

实例位置：光盘\MR\源码\第 14 章\14.7

Step 1) 修改 index.jsp 页面中的读取聊天内容的方法 getContent()，设置 get() 方法的返回数据的格式为 JSON，并将返回的 JSON 格式的聊天内容显示到页面中，修改后的代码如下：

```
function getContent() {
    $.get("ChatServlet?action=get&nocache=" + new Date().getTime(),
                    function(data) {
                            var msg="";                          //初始化聊天内容字符串
                            var chats=eval(data);
                            $.each(chats,function(i){
                                    msg+="<br>"+chats[i].message;
                                                                 //读取一条留言信息
                            });
                    $("#content").html(msg);                     //显示读取到的聊天内容
                    },"JSON");
}
```

Step 2) 修改 ChatServlet 中获取全部聊天信息的 get() 方法，将聊天内容以 JSON 格式输出，修改后的代码如下：

```
public void get(HttpServletRequest request,HttpServletResponse response)
throws ServletException,IOException{
//设置响应的内容类型及编码方式
response.setContentType("application/json;charset=UTF-8");
PrintWriter out = response.getWriter();          //获取输出流对象
out.println("[");
/********************获取聊天信息*****************************/
ServletContext application=getServletContext(); //获取 application 对象
if(null!=application.getAttribute("message")){
      Vector<String> v_temp=(Vector<String>)application.getAttribute("message");
      String msg="";
      for(int i=v_temp.size()-1;i>=0;i--){
                msg+="{\"message\":\""+v_temp.get(i)+"\"},";
      }
      out.println(msg.substring(0, msg.length()-1));//去除最后一个逗号
   }else{
      out.println("{\"message\":\"欢迎光临碧海聆音聊天室！\"}");
   }
out.println("]");
out.flush();
out.close();                                     //关闭输出流对象
}
```

运行本实例，同样可以得到如图 14.8 所示的运行结果。

14.4.5 使用 $.ajax() 方法

在 14.4.3 小节中，我们介绍了发送 GET 请求的 get() 方法和发送 POST 请求的 post() 方

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

法，虽然这两个方法可以实现发送 GET 和 POST 请求，但是这两个方法只是对请求成功的情况提供了回调函数，并未对失败的情况提供回调函数。如果需要实现对请求失败的情况提供回调函数，那么可以使用 $.ajax() 方法。$.ajax() 方法是 jQuery 中最底层的 Ajax 实现方法。使用该方法可以设置更加复杂的操作，例如，error（请求失败后处理）和 beforeSend（提前提交回调函数处理）等。使用 $.ajax() 方法，用户可以根据功能需求自定义 Ajax 操作，$.ajax() 方法的语法格式如下。

```
$.ajax( url [, settings] )
```

- url：必选参数，用于发送请求的地址（默认为当前页）。
- settings：可选参数，用于进行 Ajax 请求设置，包含许多可选的设置参数，都是以 key/value 形式体现的。常用的设置参数如表 14.4 所示。

表 14.4　settings 参数的常用设置参数

设置参数	说　明
type	用于指定请求方式，可以设置为 GET 或者 POST，默认值为 GET
data	用于指定发送到服务器的数据。如果数据不是字符串，将自动转换为请求字符串格式。在发送 GET 请求时，该数据将附加在 URL 的后面。设置 processData 参数值为 false，可以禁止自动转换。该设置参数的值必须为 key/value 格式。如果为数组，jQuery 将自动为不同值对应同一个名称。例如 {foo:["bar1", "bar2"]} 将转换为 '&foo=bar1&foo=bar2'
dataType	用于指定服务器返回数据的类型。如果不指定，jQuery 将自动根据 HTTP 包的 MIME 信息返回 responseXML 或 responseText，并作为回调函数参数传递，可用值如下： text：返回纯文本字符串 xml：返回 XML 文档，可用 jQuery 进行处理 html：返回纯文本 HTML 信息（包含的 <script> 元素会在插入 DOM 后执行） script：返回纯文本 JavaScript 代码。不会自动缓存结果，除非设置了 cache 参数 json：返回 JSON 格式的数据 jsonp：JSONP 格式。使用 JSONP 形式调用函数时，如果存在代码“url?callback=?”，那么 jQuery 将自动替换？为正确的函数名，以执行回调函数
async	设置发送请求的方式，默认是 true，为异步请求方式，同步请求方式可以设置成 false
beforeSend(jqXHR, settings)	用于设置一个发送请求前可以修改 XMLHttpRequest 对象的函数，例如，添加自定义 HTTP 头等
complete(jqXHR, textStatus)	用于设置一个请求完成后的回调函数，无论请求成功或失败，该函数均被调用
error(jqXHR, textStatus, errorThrown)	用于设置请求失败时调用的函数
success(data, textStatus, jqXHR)	用于设置请求成功时调用的函数
global	用于设置是否触发全局 AJAX 事件。设置为 true，触发全局 AJAX 事件，设置为 false 则不触发全局 AJAX 事件，默认值为 true

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（续）

设置参数	说　明
timeout	用于设置请求超时的时间（单位为毫秒）。此设置将覆盖全局设置
cache	用于设置是否从浏览器缓存中加载请求信息，设置为 true 将会从浏览器缓存中加载请求信息。默认值为 true，当 dataType 的值为 script 和 jsonp 时值为 false
dataFilter(data,type)	用于指定将 Ajax 返回的原始数据进行预处理的函数。提供 data 和 type 两个参数：data 是 Ajax 返回的原始数据，type 是调用 $.ajax() 时提供的 dataType 参数。函数返回的值将由 jQuery 进一步处理
contentType	用于设置发送信息数据至服务器时内容编码类型，默认值为 application/x-www-form-urlencoded，该默认值适用于大多数应用场合
ifModified	用于设置是否仅在服务器数据改变时获取新数据。使用 HTTP 包的 Last-Modified 头信息判断，默认值为 false

例如，在例 14.7 中，修改使用 get() 方法发送请求的代码，改为使用 $.ajax() 方法发送请求，可以使用下面的代码。

```
$.ajax({
        url : "ChatServlet",                                    // 设置请求地址
        type : "GET",                                           // 设置请求方式
        dataType : "json",                                      // 设置返回数据的类型
        data : {
                "action" : "get",
                "nocache" : new Date().getTime()
        },                                                      // 设置传递的数据
        // 设置请求成功时执行的回调函数
        success : function(data) {
                var msg = "";                                   // 初始化聊天内容字符串
                var chats = eval(data);
                $.each(chats, function(i) {
                        msg += "<br>" + chats[i].message;       // 读取一条留言信息
                });
                $("#content").html(msg);                        // 显示读取到的聊天内容
        },
        // 设置请求失败时执行的回调函数
        error : function() {
                alert("请求失败！");
        }
});
```

14.5　需要注意的几个问题

14.5.1　安全问题

安全性是互联网服务日益关注的重点，而 Web 天生就是不安全的。目前 Ajax 应用主要面临以下安全问题。

（1）JavaScript 本身的安全性

虽然 JavaScript 的安全性已逐步提高，提供了很多受限功能，包括访问浏览器的历史记

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

录、上传文件、改变菜单栏等。但是，当在 Web 浏览器中执行 JavaScript 代码时，用户允许任何人编写的代码运行在自己的机器上，这就为移动代码自动跨越网络来运行提供了方便条件，从而给网站带来了安全隐患。为了解决移动代码的潜在危险，浏览器厂商在一个 sandbox（沙箱）中执行 JavaScript 代码，沙箱是一个只能访问很少计算机资源的密闭环境，从而使 Ajax 应用不能读取或写入本地文件系统。虽然这会给程序开发带来困难，但是，它提高了客户端 JavaScript 的安全性。

说明 *移动代码是指存放在一台机器上的代码，其自身可以通过网络传输到另外一台机器执行。*

（2）数据在网络上传输的安全问题

当采用普通的 HTTP 请求时，请求参数的所有代码都是以明码的方式在网络上传输的。对于一些不太重要的数据，采用普通的 HTTP 请求即可满足要求，但是如果涉及特别机密的信息，这样做则是不行的，因为一个正常的路由不会查看传输的任何信息，而对于一个恶意的路由，则可能会读取传输的内容。为了保证 HTTP 传输数据的安全，可以对传输的数据进行加密，这样即使被看到，危险也是不大的。对传输的数据进行加密，虽然可能会降低服务器的性能，但对于敏感数据，以性能换取更高的安全，还是值得的。

（3）客户端调用远程服务的安全问题

虽然 Ajax 允许客户端完成部分服务器的工作，并可以通过 JavaScript 来检查用户的权限，但是通过客户端脚本控制权限并不可取，一些解密高手可以轻松绕过 JavaScript 的权限检查，直接访问业务逻辑组件，从而对网站造成威胁。通常情况下，在 Ajax 应用中，应该将所有的 Ajax 请求都发送到控制器，由控制器负责检查调用者是否有访问资源的权限，而所有的业务逻辑组件都隐藏在控制器的后面。

14.5.2　性能问题

由于 Ajax 将大量的计算从服务器移到了客户端，这就意味着浏览器将承受更大的负担，而不再是只负责简单的文档显示。由于 Ajax 的核心语言是 JavaScript，而 JavaScript 并不以高性能知名。另外，JavaScript 对象也不是轻量级的，特别是 DOM 元素耗费了大量的内存。因此，如何提高 JavaScript 代码的性能对于 Ajax 开发者来说尤为重要。下面介绍几种优化 Ajax 应用执行速度的方法。

- 优化 for 循环。
- 尽量使用局部变量，而不使用全局变量。
- 尽量少用 eval，每使用 eval 都需要消耗大量的时间。
- 将 DOM 节点附加到文档上。
- 尽量减少点“.”号操作符的使用。

14.5.3　浏览器兼容性问题

Ajax 使用了大量的 JavaScript 和 Ajax 引擎，而这些内容需要浏览器提供足够的支持。目前多数浏览器都支持 Ajax，除了 IE 4.0 及以下版本、Opera 7.0 及以下版本、基本文本的浏览器、没有可视化实现的浏览器以及 1997 年以前的浏览器。虽然现在我们常用的浏览器都支持

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Ajax，但是提供 XMLHttpRequest 对象的方式不一样。所以使用 Ajax 的程序必须测试针对各个浏览器的兼容性。

14.5.4 中文编码问题

Ajax 不支持多种字符集，它默认的字符集是 UTF-8，所以在应用 Ajax 技术的程序中应及时进行编码转换，否则对于程序中出现的中文字符将变成乱码。一般情况下，有以下两种情况可以产生中文乱码。

1. 发送请求时出现中文乱码

将数据提交到服务器有两种方法，一种是使用 GET 方法提交，另一种是使用 POST 方法提交。使用不同的方法提交数据，在服务器端接收参数时解决中文乱码的方法是不同的，具体解决方法如下。

1）当接收使用 GET 方法提交的数据时，要将编码转换为 GBK 或是 GB2312。例如，将省份名称的编码转换为 GBK 的代码如下：

```
String selProvince=request.getParameter("parProvince");          //获取选择的省份
selProvince=new String(selProvince.getBytes("ISO-8859-1"),"GBK");
```

说明 如果接收请求的页面的编码为 UTF-8，在接收页面则需要将接收到的数据转换为 UTF-8 编码，这时就会出现中文乱码。解决的方法是：在发送 GET 请求时，应用 encodeURIComponent() 方法对要发送的中文进行编码。

2）由于应用 POST 方法提交数据时，默认的字符编码是 UTF-8，所以当接收使用 POST 方法提交的数据时，要将编码转换为 UTF-8。例如，将用户名的编码转换为 UTF-8 的代码如下：

```
String username=request.getParameter("user");                    //获取用户名
username=new String(username.getBytes("ISO-8859-1"),"UTF-8");
```

2. 获取服务器的响应结果时出现中文乱码

由于 Ajax 在接收 responseText 或 responseXML 的值时是按照 UTF-8 的编码格式进行解码的，所以如果服务器端传递的数据不是 UTF-8 格式，在接收 responseText 或 responseXML 的值时，就可能产生乱码。解决的办法是保证从服务器端传递的数据采用 UTF-8 的编码格式。

14.6 上机实践

14.6.1 实时显示公告信息

➤➤➤ 题目描述

应用 Ajax 可以实现无刷新动态从数据库中获取实时信息。本练习尝试开发一个程序，要求应用 Ajax 实现无刷新地每隔 10 分钟从数据库获取一次最新公告，并滚动显示，效果如图 14.9 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

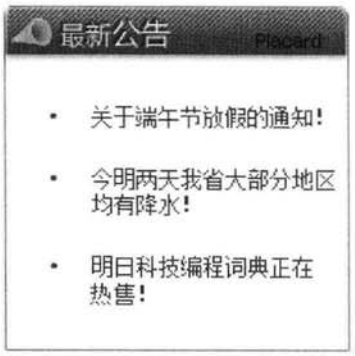


图 14.9　实时显示的公告信息

➤➤➤ 技术指导

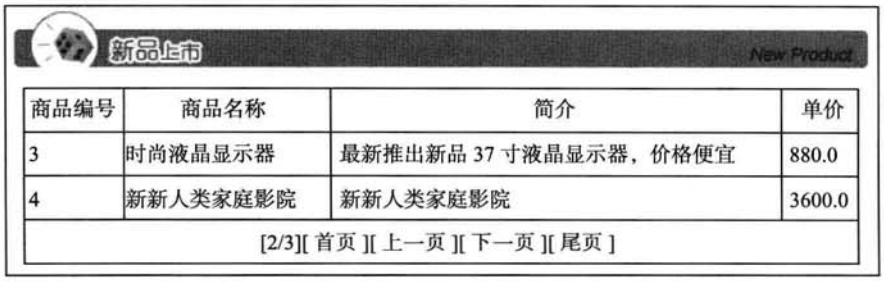
在实现本实例时，主要利用 Ajax 异步提交请求的方式来定时读取数据库中最新的公告信息。实现定时读取的关键是通过 JavaScript 中的 window 对象的 setInterval() 方法，通过该方法来调用 Ajax 请求服务器的方法，并且设置每隔 10 分钟调用一次，即可实现实时显示最新的公告信息。本上机实践的关键代码如下：

```
function getInfo(){
$.get("getInfo.jsp?nocache="+new Date().getTime(),function(data){
     $("#showInfo").html(data);
   });
}
```

14.6.2　实现无刷新分页

➤➤➤ 题目描述

在 Web 应用开发中，经常需要对信息进行分页显示。本上机实践尝试应用 Ajax 实现无刷新分页，效果如图 14.10 所示。单击信息列表下面的“首页”、“上一页”、“下一页”以及“尾页”超链接时，可以显示不同的信息，此时整个页面不刷新。



商品编号	商品名称	简介	单价
3	时尚液晶显示器	最新推出新品 37 寸液晶显示器，价格便宜	880.0
4	新新人类家庭影院	新新人类家庭影院	3600.0
[2/3][首页][上一页][下一页][尾页]			

图 14.10　实现无刷新分页

➤➤➤ 技术指导

本实例主要应用 Ajax 异步提交来实现。当单击超链接时，需要将当前页的页码参数值通过 Ajax 请求方法发送到服务器，在服务器中根据获得的参数值获取不同的分页数据，然后将

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

这些分页数据以 XML 文档格式响应给客户端页面。客户端通过 Ajax 回调函数来读取 XML 文档内容，将这些内容替换页面信息列表中原来的显示内容，而不改变页面其他内容，从而实现页面的无刷新分页。

14.6.3 多级联动下拉列表

➤➤➤ 题目描述

多级联动下拉列表是指一组相互关联的下拉列表，相邻的两个下拉列表是父子关系，改变父下拉列表的值，子下拉列表也随之改变。本练习尝试开发一个程序，要求在页面中将显示一个三级联动下拉列表，在省和直辖市的下拉列表中选择省份，在地级市下拉列表中将显示出该省份的地级市信息，在地级市下拉列表中选择其中一个市，在县、县级市或区下拉列表中将显示区信息，效果如图 14.11 所示。

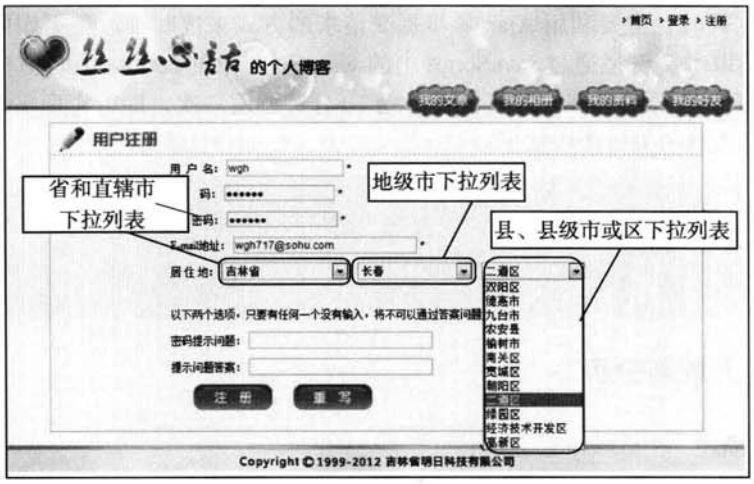


图 14.11 多级联动下拉列表

➤➤➤ 技术指导

在实现本范例时，首先需要设计一个保存省市信息的 XML 文件，在该文件中，需要充分体现出省 / 直辖市、地级市和县 / 县级市 / 区之间的关系，接下来需要从 XML 文件中获取所需的信息，也就是解析 XML 文件。本实例中应用的是 dom4j 组件来解析 XML 文件。

14.7 小结

本章首先介绍了 Ajax、Ajax 开发模式与传统开发模式的区别以及 Ajax 的技术特点，然后详细介绍了如何使用 XMLHttpRequest 对象。其中，XMLHttpRequest 对象是 Ajax 的核心技术，需要重点掌握。接下来介绍了传统 Ajax 的工作流程，主要是发送请求和处理服务器响应。最后介绍了应用 jQuery 实现 Ajax 和 Ajax 开发中需要注意的几个问题。需要读者重点掌握的是应用 jQuery 实现 Ajax，因为在以后的项目开发中比较常用。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

14.8　习题

选择题

1．下面几项技术中，Ajax 不包括的是（　　）。

A. HttpRequest 对象　　B. JavaScript　　C. XML

2．XMLHttpRequest 对象的 open() 方法中的 method 参数一般取值为（　　）。

A. get 或 post　　B. session　　C. request

3．下面哪项不属于 XMLHttpRequest 对象的常用属性（　　）。

A. onreadystatechange　　B. responseXML　　C. XML

4．XMLHttpRequest 对象是由哪个公司率先推出的（　　）。

A. 微软　　B. Sun　　C. oracle

5．要实现设置进行异步请求目标的 URL，可使用 XMLHttpRequest 对象的（　　）。

A. open() 方法　　B. send() 方法　　C. setRequestHeader() 方法

填空题

1．Ajax 是_____的缩写。

2．Ajax 是_____等多种技术的组合。

3．XMLHttpRequest 对象的_____方法用于停止当前异步请求。

4．XMLHttpRequest 对象的 readyState 属性的可选值为_____、_____、_____、_____、_____。

5．Ajax 发送请求包括 _____ 和 _____ 两种。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 15 章　MVC 架构实现者——Struts 2 框架

视频讲解：93 分钟

Strtus 2 是 Apache 软件组织的一个开放源代码项目，是基于 WebWork 核心思想的全新框架，在 Java Web 开发领域之中占有十分重要的地位。随着 JSP 技术的成熟，越来越多的开发人员专注于 MVC 框架，Struts 2 受到了广泛的青睐。本章将对 Struts 2 框架进行详细的介绍。

通过学习本章，读者可以达到以下学习目的：

- 了解 Struts 2 基础知识；
- 熟悉 Action 对象；
- 掌握 Struts 2 的配置文件、开发模式和标签库；
- 熟练使用拦截器（Interceptor）；
- 掌握数据验证机制。

15.1　Struts 2框架概述

Struts 2 框架起源于 WebWork 框架，也是一个 MVC 框架。下面将对 MVC 原理、Struts 2 框架的产生及其结构体系进行介绍。

15.1.1　理解 MVC 的原理

MVC 是一种程序设计理念，目前在 Java Web 应用中常用的流行框架有 Struts、JSF、Tapestry 和 Spring MVC 等，其中 Struts 框架的应用最为广泛。

到目前为止，Struts 框架拥有两个主要的版本，分别为 Struts 1.x 与 Struts 2.x 版本，它们都是遵循 MVC 设计理念的开源 Web 框架。在 2001 年 6 月发布的 Struts 1 版本是基于 MVC 设计理念而开发的 Java Web 应用框架，其 MVC 架构如图 15.1 所示。

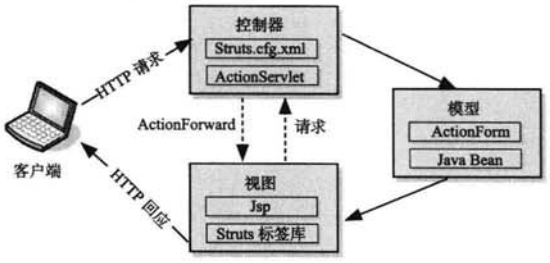


图 15.1　Struts 1 的 MVC 架构

在 Struts 1 的 MVC 架构中各层的功能如下。

（1）控制器

在 Struts 1 的 MVC 架构中，使用中央控制器 ActionServlet 充当控制层，将请求分发配置在配置文件 struts.cfg.xml 文件中。当客户端发送一个 HTTP 请求时，将由 Struts 的中央控制器分发处理请求。处理后返回 ActionForward 对象将请求转发到指定的 JSP 页面回应客户端。

（2）模型

模型层主要由 Struts 中的 ActionForm 对象及业务 Java Bean 实现，其中前者封装表单数据，能够与网页表单交互并传递数据；后者用于处理实际的业务请求，由 Action 调用。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（3）视图

视图指用户看到并与之交互的界面，即 Java Web 应用程序的外观。在 Struts 1 框架中 Struts 提供的标签库增强了 JSP 页面的功能，并通过该标签库与 JSP 页面实现视图层。

由于 Struts 1 的架构是真正意义上的 MVC 架构模式，所以在其发布以后受到了广大开发人员的认可，在 Java Web 开发领域之中 Struts 1 拥有大量的用户。

15.1.2　Struts 2 框架的产生

性能高效、松耦合和低侵入是开发人员追求的理想状态，针对 Struts 1 框架中存在的缺陷与不足，诞生了全新的 Struts 2 框架。它修改了 Struts 1 框架中的缺陷，而且还提供了更加灵活与强大的功能。

Struts 2 的结构体系与 Struts 1 有很大区别，因为该框架是在 WebWork 框架的基础上发展而来的，所以是 WebWork 技术与 Struts 技术的结合。在 Struts 的官方网站上可以看到 Struts 2 的图片，如图 15.2 所示。



图 15.2　Struts 2 的图片

WebWork 是开源组织 OpenSymphony 上一个非常优秀的开源 Web 框架，在 2002 年 3 月发布。相对于 Struts 1，其设计思想更加超前，功能也更加灵活。其中 Action 对象不再与 Servlet API 相耦合，它可以在脱离 Web 容器的情况下运行。而且 WebWork 还提供了自己的 IOC 容器，增强了程序的灵活性，通过控制反转使程序测试更加简单。由于 Struts 1 框架与 WebWork 都是非常优秀的框架，而 Struts 2 又吸收了二者的优势，因此 Struts 2 框架的前景非常美好。

15.1.3　Struts 2 的结构体系

Struts 2 是基于 WebWork 技术开发的全新 Web 框架，其结构体系如图 15.3 所示。

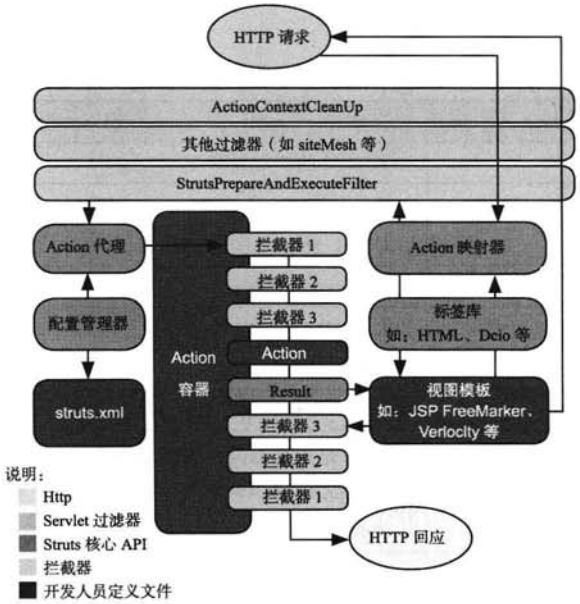


图 15.3　Struts 2 的结构体系

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Struts 2 通过过滤器拦截要处理的请求，当客户端发送一个 HTTP 请求时，需要经过一个过滤器链。这个过滤器链包括 ActionContextClearUp 过滤器、其他 Web 应用过滤器及 StrutsPrepareAndExecuteFilter 过滤器，其中 StrutsPrepareAndExecuteFilter 过滤器是必须配置的。

当 StrutsPrepareAndExecuteFilter 过滤器被调用时，Action 映射器将查找需要调用的 Action 对象，并返回该对象的代理。然后 Action 代理将从配置管理器中读取 Struts 2 的相关配置（struts.xml）。Action 容器调用指定的 Action 对象，在调用之前需要经过 Struts 2 的一系列拦截器。拦截器与过滤器的原理相似，从图中可以看出两次执行顺序是相反的。

当 Action 处理请求后，将返回相应的结果视图（JSP 和 FreeMarker 等），在这些视图之中可以使用 Struts 标签显示数据并控制数据逻辑。然后 HTTP 请求回应给浏览器，在回应的过滤中同样经过过滤器链。

15.2 Struts 2 入门

Struts 2 的使用比起 Struts 1.x 更为简单方便，只要加载一些 jar 包等插件。而不需要配置任何文件，即 Struts 2 采用热部署方式注册插件。

15.2.1 获取与配置 Struts 2

Struts 的官方网站是 http://struts.apache.org，在此网站上可以获取 Struts 的所有版本及帮助文档，本书所使用的 Struts 2 开发包为 Struts 2.3.4 版本。

在项目开发之前需要添加 Struts 2 的类库支持，即将 lib 目录中的 jar 包文件配置到项目的构建路径中。通常情况下不用全部添加这些 jar 包文件，根据项目实际的开发需要添加即可。

表 15.1 所示为开发 Struts 2 项目需要添加的类库文件，在 Struts 2.3 程序中这些 jar 文件是必须要添加的。

表 15.1 开发 Struts 2 项目需要添加的类库文件

名　称	说　明
struts2-core-2.3.4.jar	Struts 2 的核心类库
xwork-core-2.3.4.jar	xwork 的核心类库
ognl-3.0.5.jar	OGNL 表达式语言类库
freemarker-2.3.19.jar	Freemarker 模板语言支持类库
commons-io-2.0.1.jar	处理 IO 操作的工具类库
commons-fileupload-1.2.2.jar	文件上传支持类库
javassist-3.11.0.GA.jar	分析、编辑和创建 Java 字节码的类库
asm-commons-3.3.jar	ASM 是一个 Java 字节码处理框架，使用它可以动态生成 stub 类和 proxy 类，在 Java 虚拟机装载类之前动态修改类的内容
asm-3.3.jar	
commons-lang3-3.1.jar	包含了一些数据类型工具类，是 java.lang.* 的扩展

在实际的项目开发中可能还需要更多的类库支持，如 Struts 2 集成的一些插件 DOJO、JFreeChar、JSON 及 JSF 等，其相关类库到 lib 目录中查找添加即可。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

15.2.2　创建第 1 个 Struts 2 程序

Struts 2 框架主要通过一个过滤器将 Struts 集成到 Web 应用中，这个过滤器对象就是 org.apache.Struts 2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter。通过它 Struts 2 即可拦截 Web 应用中的 HTTP 请求，并将这个 HTTP 请求转发到指定的 Action 处理，Action 根据处理的结果返回给客户端相应的页面。因此在 Struts 2 框架中，过滤器 StrutsPrepareAndExecuteFilter 是 Web 应用与 Struts 2 API 之间的入口，它在 Struts 2 应用中具有重要的作用。

【例 15.1】 创建 Java Web 项目并添加 Struts 2 的支持类库，通过 Struts 2 将请求转发到指定 JSP 页面。本实例应用 Struts 2 框架处理 HTTP 请求的流程如图 15.4 所示。

实例位置：光盘 \MR\ 源码 \ 第 15 章 \15.1

Step 1) 创建名为 15.1 的 Web 项目，将 Struts 2 的类库文件添加到 WEB-INF 目录中的 lib 文件夹中。由于本实例实现功能比较简单，所以只添加 Struts 2 的核心类包即可，添加的类包如图 15.5 所示。

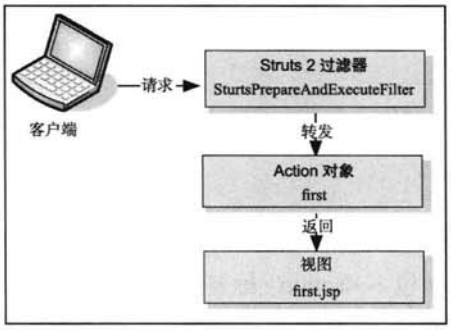


图 15.4　实例处理 HTTP 请求的流程



图 15.5　添加的类包

说明　Struts 2 的支持类库可以在下载的 Struts 2 开发包的解压缩目录中 lib 文件夹中得到。

Step 2) 在 web.xml 文件中声明 Struts 2 提供的过滤器，类名为“org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter”，其关键代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
 id="WebApp_ID" version="3.0">
  <display-name>15.1</display-name>
  <filter>                                          <!-- 配置 Struts 2 过滤器 -->
    <filter-name>struts2</filter-name>              <!-- 过滤器名称 -->
    <!-- 过滤器类 -->
    <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecute
    Filter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>struts2</filter-name>              <!-- 过滤器名称 -->
    <url-pattern>/*</url-pattern>                   <!-- 过滤器映射 -->
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    </filter-mapping>
</web-app>
```

注意 Struts 2 中使用的过滤器类为 org.apache.Struts 2.dispatcher.FilterDispatcher，从 Struts 2.1 开始已经不推荐使用了，而使用 org.apache.Struts 2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter 类。

Step 3 在 Web 项目的源码文件夹中，创建名为 struts.xml 的配置文件，在其中定义 Struts 2 中的 Action 对象，其关键代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">
<struts>
    <!-- 声明包 -->
    <package name="myPackage" extends="struts-default">
            <!-- 定义 action -->
            <action name="first">
                    <!-- 定义处理成功后的映射页面 -->
                    <result>/first.jsp</result>
            </action>
    </package>
</struts>
```

上面的代码中，<package> 标签用于声明一个包，通过 name 属性指定其名为 myPackage，并通过 extends 属性指定此包继承于 struts-default 包；<action> 标签：定义 Action 对象，其 name 属性用于指定访问此 Action 的 URL；<result> 子元素：定义处理结果和资源之间的映射关系，实例中 <result> 子元素的配置为处理成功后请求将转发到 first.jsp 页面。

说明 在 struts.xml 文件中，Struts 2 的 Action 配置需要放置在包空间内，类似 Java 中的包的概念。通过 <package> 标签声明，通常情况下声明的包需要继承于 struts-default 包。

Step 4 创建主页面 index.jsp，在其中编写一个超链接用于访问上面所定义的 Action 对象。此链接指向的地址为 first.action，其关键代码如下：

```
<body>
    <a href="first.action">请求 Struts 2</a>
</body>
```

说明 在 Struts 2 中 Action 对象的默认访问后缀为“.action”，此后缀可以任意更改，更改方法在后续内容中讲解。

Step 5 创建名为 first.jsp 的 JSP 页面作为 Action 对象 first 处理成功后的返回页面，其关键代码如下：

```
<body>
    第一个 Struts 2 程序！
</body>
```

实例运行后，打开主页面，如图 15.6 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

单击“请求 Struts 2”超链接，请求将交给 Action 对象 first 处理，在处理成功后返回如图 15.7 所示的 first.jsp 页面。

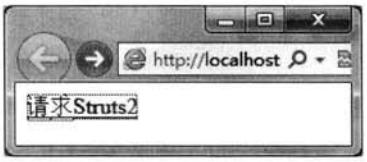


图 15.6　主页面



图 15.7　first.jsp 页面

15.3　Action对象

在传统的 MVC 框架中，Action 需要实现特定的接口，这些接口由 MVC 框架定义，实现这些接口会与 MVC 框架耦合。Struts 2 比 Action 更为灵活，可以实现或不实现 Struts 2 的接口。

15.3.1　认识 Action 对象

Action 对象是 Struts 2 框架中的重要对象，主要用于处理 HTTP 请求。在 Struts 2 API 中，Action 对象是一个接口，位于 com.opensymphony.xwork2 包中。通常情况下，我们在编写 Struts 2 项目时，创建 Action 对象都要直接或间接实现 com.opensymphony.xwork2.Action 接口，在该接口中，除了定义 execute() 方法外，还定义了 5 个字符串类型的静态常量。com.opensymphony.xwork2.Action 接口的关键代码如下：

```
    public interface Action {
    public static final String SUCCESS = "success";
    public static final String NONE = "none";
    public static final String ERROR = "error";
    public static final String INPUT = "input";
    public static final String LOGIN = "login";
    public String execute() throws Exception;
}
```

在 Action 接口中，包含了 5 个静态常量，它们是 Struts 2 API 为处理结果定义的静态常量，具体的含义如下：

（1）SUCCESS

静态变量 SUCCESS 代表 Action 执行成功的返回值，在 Action 执行成功的情况下需要返回成功页面，则可设置返回值为 SUCCESS。

（2）NONE

静态变量 NONE 代表 Action 执行成功的返回值，但不需要返回到成功页面，主要用于处理不需要返回结果页面的业务逻辑。

（3）ERROR

静态变量 ERROR 代表 Action 执行失败的返回值，在一些信息验证失败的情况下可以使 Action 返回此值。

（4）INPUT

静态变量 INPUT 代表需要返回某个输入信息页面的返回值，如修改某项信息时在加载数

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

据后需要返回到修改页面，即可将 Action 对象处理的返回值设置为 INPUT。

（5）LOGIN

静态变量 LOGIN 代表需要用户登录的返回值，如在验证用户是否登录时 Action 验证失败并需要用户重新登录，即可将 Action 对象处理的返回值设置为 LOGIN。

15.3.2 请求参数的注入原理

在 Struts 2 框架之中，表单提交的数据会自动注入到与 Action 对象中相对应的属性，它与 Spring 框架中 IOC 注入原理相同，通过 Action 对象为属性提供 setter 方法进行注入。例如，创建 UserAction 类，并提供一个 username 属性，其代码如下：

```
public class UserAction extends ActionSupport {
    private String username;                                    // 用户名属性
    // 为 username 提供 setter 方法
    public void setUsername(String username) {
            this.username = username;
    }
    // 为 username 提供 getter 方法
    public String getUsername() {
            return username;
    }
    public String execute() {
            return SUCCESS;
    }
}
```

需要注入属性值的 Action 对象必须为属性提供 setter() 方法，因为 Struts 2 的内部实现是按照 JavaBean 规范中提供的 setter 方法自动为属性注入值的。

技巧 由于 Struts 2 中 Action 对象的属性通过其 setter 方法注入，所以需要为属性提供 setter 方法。但在获取这个属性的数值时需要通过 getter 方法，因此在编写代码时最好为 Action 对象的属性提供 setter 与 getter 方法。

15.3.3 Action 的基本流程

Struts 2 框架主要通过 Struts 2 的过滤器对象拦截 HTTP 请求，然后将请求分配到指定的 Action 处理，其基本流程如图 15.8 所示。

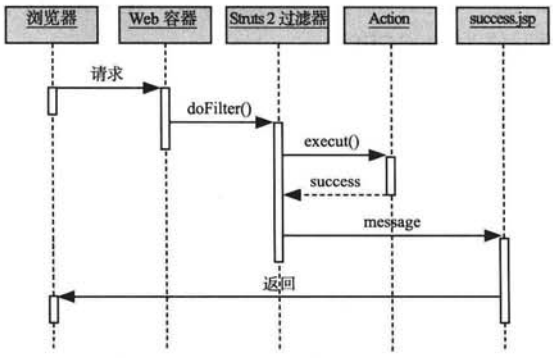


图 15.8 Struts 2 的基本流程

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

由于在 Web 项目中配置了 Struts 2 的过滤器，所以当浏览器向 Web 容器发送一个 HTTP 请求时，Web 容器就要调用 Struts 2 过滤器的 doFilter() 方法。此时 Struts 2 接收到 HTTP 请求，通过 Struts 2 的内部处理机制会判断这个请求是否与某个 Action 对象相匹配。如果找到匹配的 Action，就会调用该对象的 execute() 方法，并根据处理结果返回相应的值。然后 Struts 2 通过 Action 的返回值查找返回值所映射的页面，最后通过一定的视图回应给浏览器。

说明　在 Struts 2 框架中，一个“*.action”请求的返回视图由 Action 对象决定。其实现方法是通过查找返回的字符串对应的配置项确定返回的视图，如 Action 中的 execute() 方法返回的字符串为 success，那么 Struts 2 就会在配置文件中查找名为 success 的配置项，并返回这个配置项对应的视图。

15.3.4　动态 Action

前面所讲解的 Action 对象都是通过重写 execute() 方法处理浏览器请求，此种方式只适合比较单一的业务逻辑请求。但在实际的项目开发中业务请求的类型多种多样（如增、删、改和查一个对象的数据），如果通过创建多个 Action 对象并编写多个 execute() 方法来处理这些请求，那么不仅处理方式过于复杂，而且需要编写很多代码。当然处理这些请求的方式有多种方法，如果可以将这些处理逻辑编写在一个 Action 对象中，然后通过 execute() 方法来判断请求的是哪种业务逻辑，在判断后将请求转发到对应的业务逻辑处理方法上，这也是一种很好的解决方案。

在 Struts 2 框架中提供了 Dynamic Action 这样一个概念，称为动态 Action。通过动态请求 Action 对象中的方法实现某一业务逻辑的处理，应用动态 Action 处理方式如图 15.9 所示。

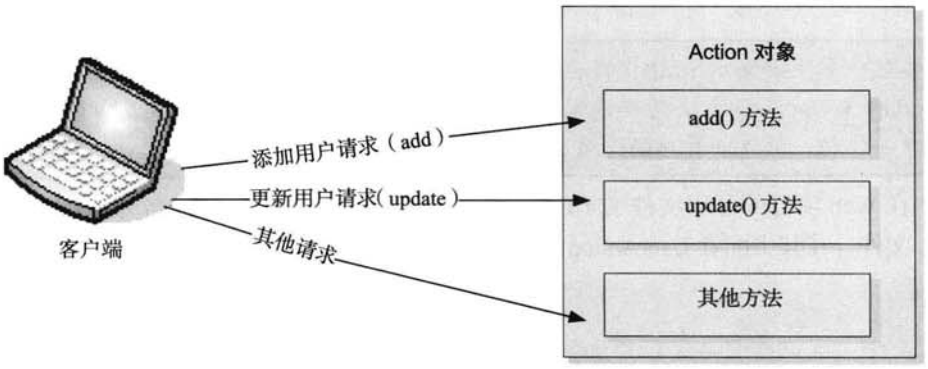


图 15.9　应用动态 Action 处理方式

从图 15.9 中可以看出动态 Action 处理方式通过请求 Action 对象中一个具体方法来实现动态操作，操作方式是通过在请求 Action 的 URL 地址后方加上请求字符串（方法名）与 Action 对象中的方法匹配，注意 Action 地址与请求字符串之间以“!”号分隔。

如在配置文件 struts.xml 文件中配置了 userAction，则请求其中的 add() 方法的格式如下：

```
/userAction!add
```

15.3.5　应用动态 Action

【例 15.2】　创建一个 Java Web 项目，应用 Struts 2 提供的动态 Action 处理添加用户信息

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

及更新用户信息请求。

实例位置：光盘\MR\源码\第15章\15.2

Step 1) 创建 Java Web 项目，将 Struts 2 的支持类库文件添加到 WEB-INF 目录中的 lib 文件夹中，然后在 web.xml 文件中注册 Struts 2 提供的过滤器。

Step 2) 创建名为 UserAction 的 Action 对象，并在其中分别编写 add() 与 update() 方法，用于处理添加用户信息及更新用户信息的请求并将请求返回到相应的页面，其关键代码如下：

```
package com.wgh;
import com.opensymphony.xwork2.ActionSupport;
public class UserAction extends ActionSupport {
    private String info;                                          // 提示信息属性
    // 添加用户信息的方法
    public String add() throws Exception {
            setInfo("添加用户信息");
            return "add";
    }
    // 修改用户信息的方法
    public String update() throws Exception {
            setInfo("修改用户信息");
            return "update";
    }
    public String getInfo() {
            return info;
    }
    public void setInfo(String info) {
            this.info = info;
    }
}
```

说明 本实例中主要演示Struts 2的动态Action处理方式，并没有实际地添加与更新用户信息。add() 与 update() 方法处理请求的方式非常简单，只为 UserAction 类中的 info 变量赋了一个值，并返回相应的结果。

Step 3) 在 Web 项目的源码文件夹（Eclipse 中默认为 src 目录）中创建名为 struts.xml 的配置文件，在其中配置 UserAction，其关键代码如下：

```
<struts>
    <!-- 声明包 -->
    <package name="user" extends="struts-default">
            <!-- 定义 action -->
            <action name="userAction" class="com.wgh.UserAction">
                    <!-- 定义处理成功后的映射页面 -->
                    <result name="add">user_add.jsp</result>
                    <result name="update">user_update.jsp</result>
            </action>
    </package>
</struts>
```

Step 4) 创建名为 user_add.jsp 的 JSP 页面作为成功添加用户信息的返回页面，其关键代码如下：

```
<body>
    <s:property value="info"/>
</body>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在 user_add.jsp 页面中，本实例通过 Struts 2 标签输出 UserAction 中的信息，即 UserAction 为中 add() 方法对 info 属性所赋的值。

Step 5 创建名为 user_update.jsp 的 JSP 页面作为成功更新用户信息的返回页面，其关键代码如下：

```
<body>
   <s:property value="info"/>
</body>
```

在 user_update.jsp 页面中，本实例通过 Struts 2 标签输出 UserAction 中的信息，即在 UserAction 中 update() 方法为 info 属性所赋的值。

Step 6 创建程序中的首页 index.jsp，在其中添加两个超链接。通过 Struts 2 提供的动态 Action 功能将这两个超链接请求分别指向于 UserAction 类的添加与更新用户信息的请求，其关键代码如下：

```
<body>
    <a href="userAction!add">添加用户 </a>
    <a href="userAction!update">修改用户 </a>
</body>
```

注意 使用 Struts 2 的动态 Action 时，在 Action 请求的 URL 地址中使用“!”号分隔 Action 请求与请求字符串，而请求字符串的名称需要与 Action 类中的方法名称相对应，否则将抛出 java.lang.NoSuchMethodException 异常。

运行实例将打开如图 15.10 所示的 index.jsp 页面，在其中显示“添加用户”与“修改用户”超链接。



图 15.10 index.jsp 页面

单击“添加用户”超链接，请求交给 UserAction 的 add() 方法处理，此时可以看到浏览器的地址栏中的地址变为 http://localhost:8080/15.2/user/userAction!add。由于使用了 Struts 2 提供的动态 Action，所以当请求 /userAction!add 时，请求会交给 UserAction 类的 add() 方法处理；当单击“修改用户”超链接后，请求将由 UserAction 类的 update() 方法处理。

技巧 从上面的实例可以出，Action 请求的处理方式并非一定要通过 execute() 方法处理，使用动态 Action 的处理方式更加方便。所以在实际的项目开发中可以将同一模块的一些请求封装在一个 Action 对象中，使用 Struts 2 提供的动态 Action 处理不同请求。

15.4 Struts 2的配置文件

在使用 Struts 2 时要配置 Struts 2 的相关文件，以使各个程序模块之间可以通信。

15.4.1 Struts 2 的配置文件类型

Struts 2 中的配置文件如表 15.2 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 15.2 Struts 2 框架的配置文件

名　称	说　明
struts-default.xml	位于 Struts 2-core-2.3.4.jar 文件的 org.apache.Struts 2 包中
struts-plugin.xml	位于 Struts 2 提供的各个插件的包中
struts.xml	Web 应用默认的 Struts 2 配置文件
struts.properties	Sturts 2 框架中属性配置文件
web.xml	此文件是 Web 应用中的 web.xml 文件，在其中也可以设置 Struts 2 框架的一些信息

其中 struts-default.xml 和 struts-plugin.xml 文件是 Struts 2 提供的配置文件，它们都在 Struts 2 提供的包中；而 struts.xml 文件是 Web 应用默认的 Struts 2 配置文件；struts.properties 文件是 Struts 2 框架中的属性配置文件，表 15.2 中后两个配置文件需要开发人员编写。

15.4.2 配置 Struts 2 包

在 struts.xml 文件中存在一个包的概念，类似 Java 中的包。配置文件 struts.xml 中的包使用 <package> 元素声明，主要用于放置一些项目中的相关配置，可以将其理解为配置文件中的一个逻辑单元。已经配置好的包可以被其他包所继承，从而提高配置文件的重用性。与 Java 中的包类似，在 struts.xml 文件中使用包不仅可以提高程序的可读性，而且还可以简化日后的维护工作，其使用方式如下：

```
<struts>
   <!-- 声明包 -->
                <package name="user" extends="struts-default">
                    ...
                </package>
</struts>
```

包使用 <package> 元素声明，必须拥有一个 name 属性来指定包的名称，<package> 元素包含的属性如表 15.3 所示。

表 15.3 <package> 元素包含的属性

属　性	说　明
name	声明包的名称，以方便在其他处引用此包，此属性是必需的
extends	用于声明继承的包，即其父包
namespace	指定名称空间，即访问此包下的 Action 需要访问的路径
abstract	将包声明为抽象类型（包中不定义 action）

15.4.3 配置名称空间

在 Java Web 开发中，Web 文件目录通常以模块划分，如用户模块的首页可以定义在“/user”目录中，其访问地址为“/user/index.jsp”。在 Struts 2 框架中，Struts 2 配置文件提供了名称空间的功能，用于指定一个 Action 对象的访问路径，它的使用方法是通过在配置文件 struts.xml 的包声明中使用 namespace 属性进行声明。

【例 15.3】 修改例 15.2 的程序，为原来的 user 包配置名称空间。

实例位置：光盘 \MR\ 源码 \ 第 15 章 \15.3

Step 1) 打开 struts.xml 文件，将 <package> 标记修改为以下内容，也就是指定名称空间为“/user”。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<package name="user" extends="struts-default" namespace="/user">
```

注意 在<package>元素中指定名称空间属性，名称空间的值需要以“/”开头，否则找不到Action对象的访问地址。

Step 2) 在项目的WebContent节点中，创建user文件夹，并将user_add.jsp和user_update.jsp文件移动到该文件夹中。修改index.jsp文件中的访问地址，在原访问地址前加上名称空间中指定的访问地址，关键代码如下：

```
<a href="user/userAction!add">添加用户</a>
<a href="user/userAction!update">修改用户</a>
```

运行本实例，将会得到与例15.2同样的运行结果。这样我们就通过配置名称空间将关于用户操作的内容放置到单独的文件夹中了。

15.4.4 Action的相关配置

Struts 2框架中的Action对象是一个控制器的角色，Struts 2框架通过它处理Http请求，其请求地址的映射需要在struts.xml文件中使用<action>元素配置，如：

```
<action name="userAction" class="com.wgh.action.UserAction" method="save">
    <result>success.jsp</result>
</action>
```

配置文件中的<action>元素主要用于建立Action对象的映射，通过该元素可以指定Action请求地址及处理后的映射页面，<action>元素常用的属性如表15.4所示。

表15.4 <action>元素的属性

属 性	说 明
name	用于配置Action对象被请求的URL映射
class	指定Action对象的类名
method	设置请求Action对象时调用该对象的哪一个方法
converter	指定Action对象类型转换器的类

注意 在<action>元素中name属性是必须配置的，在建立Action对象的映射时必须指定其URL映射地址，否则请求找不到Action对象。

在实际的项目开发之中，每一个模块的业务逻辑都比较复杂，一个Action对象可包含多个业务逻辑请求的分支。

在用户管理模块中需要对用户信息执行添加、删除、修改和查询操作，代码如下：

```
import com.opensymphony.xwork2.ActionSupport;
/**
 * 用户信息管理Action
 */
public class UserAction extends ActionSupport{
    private static final long serialVersionUID = 1L;
    // 添加用户信息
    public String save() throws Exception {
            ...
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        return SUCCESS;
    }
    // 修改用户信息
    public String update() throws Exception {
        ...
        return SUCCESS;
    }
    // 删除用户信息
    public String delete() throws Exception {
        ...
        return SUCCESS;
    }
    // 查询用户信息
    public String find() throws Exception {
        ...
        return SUCCESS;
    }
}
```

调用一个 Action 对象，默认执行的是 execute() 方法。如果在多业务逻辑分支的 Action 对象中，需要请求指定的方法，则可通过 <action> 元素的 method 属性配置。即将一个请求交给指定的业务逻辑方法处理，代码如下：

```
<!-- 添加用户 -->
<action name="userAction" class="com.lyq.action.UserAction" method="save">
    <result>success.jsp</result>
</action>
<!-- 修改用户 -->
<action name="userAction" class="com.lyq.action.UserAction" method="update">
    <result>success.jsp</result>
</action>
<!-- 删除用户 -->
<action name="userAction" class="com.lyq.action.UserAction" method="delete">
    <result>success.jsp</result>
</action>
<!-- 查询用户 -->
<action name="userAction" class="com.lyq.action.UserAction" method="find">
    <result>success.jsp</result>
</action>
```

<action> 元素的 method 属性主要用于为一个 action 请求分发一个指定业务逻辑方法，如设置为 add，那么这个请求就会交给 Action 对象的 add() 方法处理，此种配置方法可以减少 Action 对象的数目。

注意 <action> 元素的 method 属性值必须与 Action 对象中的方法名一致，这是因为 Struts 2 框架通过 method 属性值查找与其匹配的方法。

15.4.5 使用通配符简化配置

在 Struts 2 框架的配置文件 struts.xml 中支持通配符，此种配置方式主要针对多个 Action 的情况。通过一定的命名约定使用通配符来配置 Action 对象，从而达到一种简化配置的效果。

在 struts.xml 文件中，常用的通配符有如下两个。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

1）通配符“*”：匹配 0 个或多个字符。

2）通配符“\”：一个转义字符，如需要匹配“/”，则使用“\/”匹配。

【例 15.4】 在 Struts 2 框架的配置文件 struts.xml 中应用通配符，代码如下：

实例位置：光盘 \MR\ 源码 \ 第 15 章 \15.4

```
<struts>
    <package name="default" namespace="/" extends="struts-default">
        <action name="*Action" class="com.wgh.action.{1}Action">
            <result name="success">result.jsp </result>
            <result name="update">update.jsp</result>
            <result name="del">result.jsp</result>
        </action>
    </package>
</struts>
```

<action> 元素的 name 属性值为“*Action”，匹配的是以字符 Action 结尾的字符串，如 UserAction 和 BookAction。在 Struts 2 框架的配置文件中是可以使用表达式 {1}、{2} 或 {3} 的方式获取通配符所匹配的字符，如代码中的“com.wgh.action.{1}Action”。

15.4.6　配置返回结果

在 MVC 的设计思想中，处理业务逻辑之后需要返回一个视图，Struts 2 框架通过 Action 的结果映射配置返回视图。

Action 对象是 Struts 2 框架中的请求处理对象，针对不同的业务请求及处理结果返回一个字符串，即 Action 处理结果的逻辑视图名。Struts 2 框架根据逻辑视图名在配置文件 struts.xml 中查找与其匹配的视图，在找到后将这个视图回应给浏览器，如图 15.11 所示。

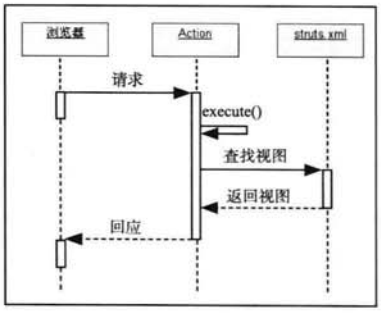


图 15.11　结果映射

在配置文件 struts.xml 文件中结果映射使用 <result> 元素，使用方法如下。

```
<action name="user" class="com.wgh.action.UserAction">
   <!-- 结果映射 -->
   <result>/user/Result.jsp</result>
   <!-- 结果映射 -->
   <result name="error">/user/Error.jsp</result>
   <!-- 结果映射 -->
   <result name="input" type="dispatcher">/user/Input.jsp</result>
</action>
```

<result> 元素的两个属性为 name 和 type，其中 name 属性用于指定 Result 的逻辑名称，与 Action 对象中方法的返回值相对应。如 execute() 方法返回值为 input，那么就将 <result> 元素的 name 属性配置为 input 对应 Action 对象返回值；type 属性用于设置返回结果的类型，如请求转发和重定向等。

注意　无 <result> 元素的 name 属性的默认值为 success。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

15.5 Struts 2的标签库

要在 JSP 中使用 Struts 2 的标签库，首先要指定标签的引入，在 JSP 代码的顶部添加以下代码：

```
<%@taglib prefix="s" url="/struts-tags" %>
```

15.5.1 应用数据标签

1. property 标签

property 标签是一个常用标签，作用是获取数据值并直接输出到页面中，其属性如表 15.5 所示。

表 15.5 property 标签的属性

名 称	说 明
default	可选
escape	可选
escapeJavaScript	可选
value	可选

2. set 标签

set 标签用于定义一个变量并为其赋值，同时设置变量的作用域（application、request 和 session）。在默认情况下，通过 set 标签定义的变量被放置到值栈中。该标签的属性如表 15.6 所示。

表 15.6 set 标签的属性

名 称	是否必须	类 型	说 明
scope	可选	String	设置变量的作用域，取值为 application、request、session、page 或 action，默认值为 action
Value	可选	String	设置变量值
var	可选	String	定义变量名

说明 在 set 标签中还包含 id 与 name 属性，在本书所讲述的 Struts 2 版本中这两个属性已过时，所以不再讲解。

set 标签的使用方式如下：

```
<s:set var="username" value="'测试 set 标签 '" scope="request"></s:set>
<s:property default="没有数据！ " value="#request.username"/>
```

上述代码通过 set 标签定义了一个名为 username 的变量，其值是一个字符串，作用域在 request 范围之中。

3. a 标签

a 标签用于构建一个超链接，最终构建效果将形成一个 HTML 中的超链接，其常用属性如表 15.7 所示。

表 15.7 a 标签的常用属性

名 称	是否必须	类 型	说 明
action	可选	String	将超链接的地址指向 action
href	可选	String	超链接地址

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（续）

名　称	是否必须	类　型	说　明
id	可选	String	设置 HTML 中的属性名称
method	可选	String	如果超链接的地址指向 action，method 同时可以为 action 声明所调用的方法
namespace	可选	String	如果超链接的地址指向 action，namespace 可以为 action 声明名称空间

4．param 标签

param 标签用于为参数赋值，可以作为其他标签的子标签。该标签的属性如表 15.8 所示。

表 15.8　param 标签的属性

名　称	是否必须	类　型	说　明
name	可选	String	设置参数名称
value	可选	Object	设置参数值

5．action 标签

action 标签是一个常用的标签，用于执行一个 Action 请求。当在一个 JSP 页面中通过 action 标签执行 Action 请求时，可以将其返回结果输出到当前页面中，也可以不输出，其常用属性如表 15.9 所示。

表 15.9　action 标签的常用属性

名　称	是否必须	类　型	说　明
executeResult	可选	boolean	是否使 Action 返回执行结果，默认值为 false
flush	可选	boolean	输出结果是否刷新，默认值为 true
ignoreContextParams	可选	boolean	是否将页面请求参数传入被调用的 Action，默认值为 false
name	必须	String	Action 对象映射的名称，即 struts.xml 中配置的名称
namespace	可选	String	指定名称空间的名称
var	可选	String	引用此 action 的名称

6．push 标签

push 标签用于将对象或值压入到值栈中并放置在顶部，因为值栈中的对象可以直接调用，所以该标签主要作用是简化操作。Push 标签的属性只有 value，用于声明压入值栈中的对象，该标签的使用方法如下。

```
<s:push value="#request.student"></s:push>
```

7．date 标签

date 标签用于格式化日期时间，可以通过指定的格式化样式格式化日期时间值，该标签的属性如表 15.10 所示。

表 15.10　date 标签的属性

名　称	是否必须	类　型	说　明
format	可选	String	设置格式化日期的样式
name	必须	String	日期值
nice	可选	boolean	是否输出给定日期与当前日期之的间时差，默认值为 false，不输出时差
var	可选	String	格式化时间的名称变量，通过此变量可以对其进行引用

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

8．include 标签

include 标签的作用类似 JSP 中的 <include> 动作标签，用于包含一个页面，并且可以通过 param 标签向目标页面中传递请求参数。

include 标签只有一个必选的 file 属性，用于包含一个 JSP 页面或 Servlet，其使用方法如下：

```
<%@include file=" /pages/common/common_admin.jsp"%>
```

9．url 标签

url 标签中提供了多个属性满足不同格式的 URL 需求，其常用属性如表 15.11 所示。

表 15.11　url 标签的常用属性

名　称	是否必须	类　型	说　明
action	可选	String	Action 对象的映射 URL，即对象的访问地址
anchor	可选	String	此 URL 的锚点
encode	可选	boolean	是否编码参数，默认值为 true
escapeAmp	可选	String	是否将“&”转义为“&amp”
forceAddSchemeHostAndPort	可选	boolean	是否添加 URL 的主机地址及端口号，默认值为 false
includeContext	可选	boolean	生成的 URL 是否包含上下文路径，默认值为 true
includeParams	可选	String	是否包含可选参数，可选值为 none、get 和 all，默认值为 none
method	可选	String	指定请求 Action 对象所调用的方法
namespace	可选	String	指定请求 Action 对象映射地址的名称空间
scheme	可选	String	指定生成 URL 所使用的协议
value	可选	String	指定生成 URL 的地址值
var	可选	String	定义生成 URL 变量名称，可以通过此名称引用 URL

url 标签是一个常用的标签，在其中可以为 url 传递请求参数，也可以通过该标签提供的属性生成不同格式的 url。

15.5.2　应用控制标签

1．if 标签

if 标签是一个流程控制标签，用于处理某一逻辑的多种条件。通常表现为“如果满足某种条件，则执行某种处理，否则执行另一种处理”。

2．<s:if> 标签

该标签是基本流程控制标签，用于在满足某个条件下的情况下执行标签体中的内容，可以单独使用。

3．<s:elseif> 标签

此标签需要与 <s:if> 标签配合使用，在不满足 <s:if> 标签中条件的情况下，判断是否满足

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



<s:elseif> 标签中的条件，如果满足，那么将执行其标签体中的内容。

4. <s:else> 标签

此标签需要与 <s:if> 或 <s:elseif> 标签配合使用，在不满足所有条件的情况下，可以使用 <s:else> 标签来执行其中的代码。

与 Java 语言相同，Struts 2 框架的流程控制标签同样支持 if…else if…else 的条件语句判断，使用方法如下。

```
<s:if test="表达式（布尔值）">
    输出结果…
</s:if>
<s:elseif test="表达式（布尔值）">
    输出结果…
</s:elseif>
可以使用多个 <s:elseif>
...
<s:else>
    输出结果…
</s:else>
```

<s:if> 与 <s:elseif> 标签都有一个名为 test 的属性，用于设置标签的判断条件，其值是一个布尔类型的条件表达式。在上述代码中可以包含多个 <s:elseif> 标签，针对不同的条件执行不同的处理。

5. iterator 标签

iterator 标签是一个迭代数据的标签，可以根据循环条件遍历数组和集合类中的所有或部分数据，并迭代出集合或数组的所有数据，也可以指定迭代数据的起始位置、步长及终止位置来迭代集合或数组中的部分数据。该标签的属性如表 15.12 所示。

表 15.12　iterator 标签的属性

名　称	是否必须	类　型	说　明
begin	可选	Integer	指定迭代数组或集合的起始位置，默认值为 0
end	可选	Integer	指定迭代数组或集合的结束位置，默认值为集合或数组的长度
status	可选	String	迭代过程中的状态
step	可选	Integer	设置迭代的步长，如果指定此值，则每一次迭代后索引值将在原索引值的基础上增加 step 值，默认值为 1
value	可选	String	指定迭代的集合或数组对象
var	可选	String	设置迭代元素的变量，如果指定此属性，那么所迭代的变量将压入到值栈之中

status 属性用于获取迭代过程中的状态信息，在 Struts 2 框架的内部结构中该属性实质是获取了 Struts 2 封装的一个迭代状态的 org.apache.Struts2.views.jsp.IteratorStatus 对象，通过此对象可以获取迭代过程中的如下信息。

（1）元素数

IteratorStatus 对象提供了 getCount() 方法来获取迭代集合或数组的元素数，如果 status 属

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

性设置为 st，那么可通过 st.count 获取元素数。

（2）是否为第 1 个元素

IteratorStatus 对象提供了 isFirst() 方法来判断当前元素是否为第 1 个元素，如果 status 属性设置为 st，那么可通过 st.first 判断当前元素是否为第 1 个元素。

（3）是否为最后一个元素

IteratorStatus 对象提供了 isLast() 方法来判断当前元素是否为最后一个元素，如果 status 属性设置为 st，那么可通过 st.last 判断当前元素是否为最后一个元素。

（4）当前索引值

IteratorStatus 对象提供了 getIndex() 方法来获取迭代集合或数组的当前索引值，如果 status 属性设置为 st，那么可通过 st.index 获取当前索引值。

（5）索引值是否为偶数

IteratorStatus 对象提供了 isEven() 方法来判断当前索引值是否为偶数，如果 status 属性设置为 st，那么可通过 st.even 判断当前索引值是否为偶数。

（6）索引值是否为奇数

IteratorStatus 对象提供了 isOdd() 方法来判断当前索引值是否为奇数，如果 status 属性设置为 st，那么可通过 st.odd 判断当前索引值是否为奇数。

15.5.3 应用表单标签

在 Struts 2 框架中 Struts 2 提供了一套表单标签，用于生成表单及其中的元素，如文本框、密码框和选择框等。它们能够与 Struts 2 API 很好地交互，常用的表单标签如表 15.13 所示。

表 15.13 常用的表单标签

名　称	说　明
form 标签	用于生成一个 form 表单
hidden 标签	用于生成一个 HTML 中的隐藏表单元素，相当于使用了 HTML 代码 <input type="hidden">
textfield 标签	用于生成一个 HTML 中的文本框元素，相当于使用了 HTML 代码 <input type="text">
password 标签	用于生成一个 HTML 中的密码框元素，相当于使用了 HTML 代码 <input type="password">
radio 标签	用于生成一个 HTML 中的单选按钮元素，相当于使用了 HTML 代码 <input type="radio">
select 标签	用于生成一个 HTML 中的下拉列表元素，相当于使用了 HTML 代码 <select><option></option> </select>
textarea 标签	用于生成一个 HTML 中的文本域元素，相当于使用了 HTML 代码 <textarea></textarea>
checkbox 标签	用于生成一个 HTML 中的选择框元素，相当于使用了 HTML 代码 <input type="checkbox">
checkboxlist 标签	用于生成一个或多个 HTML 中的选择框元素，相当于使用了 HTML 代码 <input type="checkboxlist">
submit 标签	用于生成一个 HTML 中的提交按钮元素，相当于使用了 HTML 代码 <input type="submit">
reset 标签	用于生成一个 HTML 中的重置按钮元素，相当于使用了 HTML 代码 <input type="reset">

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表单标签的常用属性如表 15.14 所示。

表 15.14　表单标签的常用属性

名　称	说　明
name	指定表单元素的 name 属性
title	指定表单元素的 title 属性
cssStyle	指定表单元素的 style 属性
cssClass	指定表单元素的 class 属性
required	用于在 lable 上添加“*”号，其值为布尔类型。如果为 true，则添加“*”号；否则不添加
disable	指定表单元素的 disable 属性
value	指定表单元素的 value 属性
labelposition	用于指定表单元素 label 的位置，默认值为 left
requireposition	用于指定在表单元素 label 上添加“*”号的位置，默认值为 right

主题是 Struts 2 框架提供的一项功能，可以为 Struts 2 框架中的表单与 UI 标签设置主题样式。默认情况下，Struts 2 提供了如下 4 种主题样式。

（1）simple

simple 主题的功能较弱，只提供了简单的 HTML 输出。

（2）xhtml

xhtml 主题是在 simple 上的扩展，它提供了简单的布局样式，可以将元素用到表格布局中，并且提供了 lable 的支持。

（3）css_xhtml

css_xhtml 主题是在 xhtml 主题基础上的扩展，它在功能上强化了 xhtml 主题在 CSS 上的样式的控制。

（4）ajax

ajax 主题是在 css_xhtml 主题上扩展，它在功能上主要强化了 css_xhtml 主题在 Ajax 方面的应用。

说明　默认情况下，Struts 2 框架应用 xhtml 主题。

xhtml 主题使用固定样式设置，非常不方便。如果不希望直接使用 HTML 来设计页面中的主题，则可以应用 simple 主题样式。

说明　Struts 2 框架的主题样式基于模板语言设计，要求开发人员了解模板语言，目前其应用并不广泛。

15.6　Struts 2的开发模式

15.6.1　实现与 Servlet API 的交互

Struts 2 中提供了 Map 类型的 request、session 与 application，可以从 ActionContext 对象中获得。该对象位于 com.opensymphony.xwork2 包中，是 Action 执行的上下文，其常用 API 方法如下。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（1）实例化 ActionContext

在 Struts 2 的 API 中，ActionContext 的构造方法需要传递一个 Map 类的上下文对象，应用这个构造方法创建 ActionContext 对象非常不方便。所以通常情况下使用该对象提供的 getContext() 方法创建，其方法声明如下：

```
public static ActionContext getContext()
```

该方法是一个静态方法，可以直接调用，其返回值是 ActionContext。

（2）获取 Map 类型的 request

获取 Struts 2 封装的 Map 类型的 request，使用 ActionContext 对象提供的 get() 方法，其方法声明如下：

```
public Object get(Object key)
```

该方法的入口参数为 Object 类型的值，获取 request 可以将其设置为 request，如：

```
Map request = ActionContext.getContext.get("request");
```

说明 ActionContext 对象提供的 get() 方法也可以获取 session 及 local 等对象。

（3）获取 Map 类型的 session

ActionContext 提供了一个直接获取 session 的方法 getSession()，其方法声明如下：

```
public Map getSession()
```

该方法返回 Map 对象，它将作用于 HttpSession 范围中。

（4）获取 Map 类型的 application

ActionContext 对象为获取 Map 类型的 application 提供了单独的 getApplication() 方法，其方法声明如下：

```
public Map getApplication()
```

该方法返回 Map 对象，作用于 ServletContext 范围中。

15.6.2 域模型 DomainModel

在讲述前面的内容时，无论用户注册逻辑，还是其他一些表单信息的提交操作均未通过操作实际的域对象实现，原因是将所有的实体对象的属性都封装在了 Action 对象中。而 Action 对象只是操作一个实体对象中的属性，不操作某一个实体对象，这样的操作有些偏离了域模型设计的思想。比较好的设计是将某一领域的实体直接封装为一个实体对象，如操作用户信息可以将用户信息封装为一个域对象 User，并将用户所属的组封装为 Group 对象，如图 15.12 所示。

将一些属性信息封装为一个实体对象的优点很多，如将一个用户信息数据保存在数据库中只需要传递一个 User 对象，而不是传递多个属性。在 Struts 2 框架中提供了操作域对象的方法，不仅可以在 Action 对象中引用某一个实体对象（见图 15.13），而且可以将 Http 请求中的参数值注入到实体对象的属性中，这种方式即 Struts 2 提供的使用 Domain Model 的方式。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

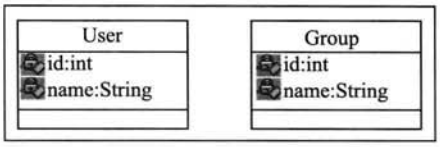


图 15.12　域对象

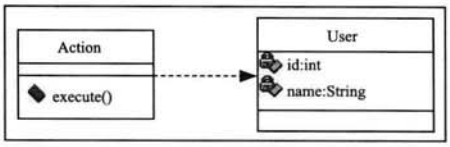


图 15.13　Action 对象引用 User 对象

例如，在 Action 中应用一个 User 对象的代码如下：

```
public class UserAction extends ActionSupport {
 private User user;
 @Override
 public String execute() throws Exception {
       return SUCCESS;
 }
 public User getUser() {
       return user;
 }
 public void setUser(User user) {
       this.user = user;
 }
}
```

在页面中提交注册请求的代码如下：

```
<body>
 <h2>用户注册</h2>
 <s:form action="userAction" method="post">
       <s:textfield name="user.name" label="用户名"></s:textfield>
       <s:password name="user.password" label="密码" ></s:password>
       <s:radio name="user.sex" list="#{1 : '男', 0 : '女'}" label="性别" ></s:radio>
       <s:submit value="注册"></s:submit>
 </s:form>
</body>
```

15.6.3　驱动模型 ModelDriven

在 Domain Model 模式中，虽然 Struts 2 的 Action 对象可以通过直接定义实例对象的引用来调用实体对象执行相关操作，但要求请求参数必须指定参数对应的实体对象。如在表单中需要指定参数名为 user.name，此种做法还是有一些不方便。Struts 2 框架还提供了另外一种方式 ModelDriven，不需要指定请求参数所属的对象引用，即可向实体对象中注入参数值。

在 Struts 2 框架的 API 中提供了一个名为 ModelDriven 的接口，Action 对象可以通过实现此接口获取指定的实体对象。获取方式是实现该接口提供的 getModel() 方法，其语法格式如下：

```
T getModel();
```

说明　ModelDriven 接口应用了泛型，getModel 的返回值为要获取的实体对象。

如果 Action 对象实现了 ModelDriven 接口，当表单提交到 Action 对象后其处理流程如图 15.14 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

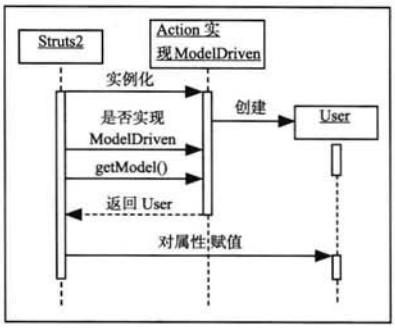


图 15.14　处理流程

Struts 2 首先实例化 Action 对象，然后判断该对象是否是 ModelDriven 对象（是否实现了 ModelDriven 接口），如果是，则调用 getModel() 方法来获取实体对象模型，并将其返回（如图中调用的 User 对象）。在之后的操作中已经存在明确的实体对象，所以不用在表单中的元素名称上添加指定实例对象的引用名称。

例如，应用以下代码添加表单。

```
<s:form action="userAction" method="post">
 <s:textfield name="name" label="用户名"></s:textfield>
 <s:password name="password" label="密码" ></s:password>
 <s:radio name="sex" list="#{1 : '男', 0 : '女'}" label="性别" ></s:radio>
 <s:submit value="注册"></s:submit>
</s:form>
```

那么处理表单请求的 UserAction 对象，同时需要实现 ModelDriven 接口及其 getModel() 方法，返回明确的实体对象 user，UserAction 类的关键代码如下：

```
public class UserAction extends ActionSupport implements ModelDriven<User> {
 private User user = new User();
 /**
   * 请求处理方法
   */
 @Override
 public String execute() throws Exception {
      return SUCCESS;
 }
 @Override
 public User getModel() {
      return this.user;
 }
}
```

由于 UserAction 实现了 ModelDriven 接口，getModel() 方法返回明确的实体对象 user，所以表单中的元素名称不用指定明确的实体对象引用即可成功地将表单提交的参数注入到 user 对象中。

说明　UserAction 类中的 user 属性需要初始化，否则在 getModel() 方法获取实体对象时，将出现空指针异常。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

15.7　Struts 2的拦截器

拦截器其实是AOP（面向方面编程）的一种实现方式，通过它可以在Action执行前后处理一些相应的操作。Struts 2提供了多个拦截器，开发人员也可以根据需要配置拦截器。

15.7.1　拦截器概述

拦截器是Struts 2框架中的一个重要的核心对象，它可以动态增强Action对象的功能，Struts 2框架中很多重要的功能都是通过拦截器实现。如在使用Struts 2框架时我们发现Struts 2与Servlet API解耦，Action对请求的处理不依赖于Servlet API，但Struts 2的Action却具有更加强大的请求处理功能。这个功能的实现就是拦截器对Action的增强，可见拦截器的重要性。此外，Struts 2框架中的表单重复提交、对象类型转换、文件上传，还有前面所学习的ModelDriven的操作都离不开拦截器的幕后操作。Struts 2的拦截器的处理机制是Struts 2框架的核心。

拦截器动态作用于Action与Result之间，可以动态地增强Action及Result（在其中添加新功能），如图15.15所示。

客户端发送的请求会被Struts 2的过滤器所拦截，此时Struts 2对请求持有控制权。它会创建Action的代理对象，并通过一系列拦截器处理请求，最后交给指定的Action处理。在这一期间，拦截器对象作用Action和Result的前后可以执行任何操作，所以Action对象编写简单是由于拦截器进行了处理。拦截器操作Action对象的顺序如图15.16所示。

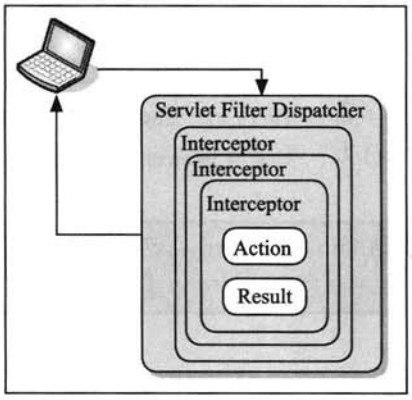


图15.15　拦截器

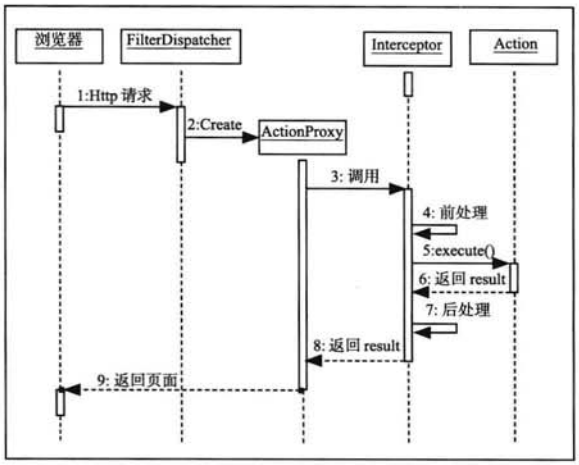


图15.16　拦截器操作Action对象的顺序

当浏览器在请求一个Action时会经过Struts 2框架的入口对象——Struts 2过滤器，此时该过滤器会创建Action的代理对象。之后通过拦截器即可在Action对象执行前后执行一些操作，如图中的“前处理”与“后处理”，最后返回结果。

15.7.2　拦截器API

在Struts 2 API中有一个名为com.opensymphony.xwork2.interceptor的包，其中有一些

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Struts 2 内置的拦截器对象，它们具有不同的功能。在这些对象中，Interceptor 接口是 Struts 2 框架中定义的拦截器对象，其他拦截器都直接或间接地实现于此接口。

在拦截器 Interceptor 中包含了 3 个方法，其代码如下：

```
public interface Interceptor extends Serializable {
 void destroy();
 void init();
 String intercept(ActionInvocation invocation) throws Exception;
}
```

destroy() 方法指示拦截器的生命周期结束，它在拦截器被销毁前调用，用于释放拦截器在初始化时占用的一些资源。

init() 方法用于对拦截器执行一些初始化操作，此方法在拦截器被实例化后和 intercept() 方法执行前调用。

intercept() 方法是拦截器中的主要方法，用于执行 Action 对象中的请求处理方法及其前后的一些操作，动态增强 Action 的功能。

说明 只有调用了 intercept() 方法中 invocation 参数的 invoke() 方法，才可以执行 Action 对象中的请求处理方法。

虽然 Struts 2 提供了拦截器对象 Interceptor，但此对象是一个接口。如果通过此接口创建拦截器对象，则需要实现 Interceptor 提供的 3 个方法。在实际开发中主要用到 intercept() 方法，如果要空实现没有用到 init() 与 destroy() 方法，这种创建拦截器方式似乎有一些不便。

为了简化程序开发，也可以通过 Struts 2 API 中的 AbstractInterceptor 对象创建拦截器对象，它与 Interceptor 接口的关系如图 15.17 所示。

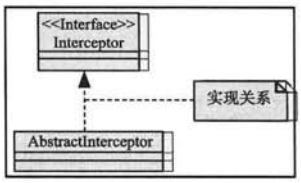


图 15.17 AbstractInterceptor 对象与 Interceptor 接口的关系

AbstractInterceptor 对象是一个抽象类，实现了 Interceptor 接口，在创建拦截器时可以通过继承该对象创建。在继承 AbstractInterceptor 对象后，创建拦截器的方式更加简单，除了重写必须的 intercept() 方法外，如果没有用到 init() 与 destroy() 方法，则不必实现。

说明 AbstractInterceptor 对象已经实现了 Interceptor 接口的 init() 与 destroy() 方法，所以通过继承该对象创建拦截器，则不需要实现这两个方法。如果需要，可以重写。

15.7.3 使用拦截器

如果在 Struts 2 框架中创建了一个拦截器对象，则配置后才可以应用到 Action 对象，配置时使用 <interceptor-ref> 标签。

【例 15.5】 为 Action 对象配置输出执行时间的拦截器，以查看执行 Action 所需的时间。

实例位置：光盘 \MR\ 源码 \ 第 15 章 \15.5

Step 1) 创建动态的 Java Web 项目，将 Struts 2 的相关类包配置到构建路径中，并在 web.xml 文件中注册 Struts 2 提供的 StrutsPrepareAndExecuteFilter 过滤器，从而搭建 Struts 2 的开发环境。

Step 2) 创建名为 TestAction 的类，此类继承于 ActionSupport 对象，其关键代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
public class TestAction extends ActionSupport {
 private static final long serialVersionUID = 1L;
 public String execute() throws Exception{
        // 线程睡眠1秒
        Thread.sleep(1000);
        return SUCCESS;
 }
}
```

说明　由于实例需要配置输出 Action 执行时间的拦截器，为了方便查看执行时间，在 execute() 方法中通过 Thread 类的 sleep() 方法使当前线程睡眠 1 秒钟。

Step 3　在 struts.xml 配置文件中配置 TestAction 对象，并将输出 Action 执行时间的拦截器 timer 应用到 TestAction 中，其关键代码如下：

```
<struts>
 <!-- 声明常量（开发模式） -->
 <constant name="struts.devMode" value="true" />
 <!-- 声明常量（在 Struts 2 的配置文件修改后自动加载） -->
 <constant name="struts.configuration.xml.reload" value="true" />
 <!-- 声明包 -->
 <package name="myPackge" extends="struts-default">
        <!-- 配置 Action -->
        <action name="testAction" class="com.lyq.action.TestAction">
            <!-- 配置拦截器 -->
            <interceptor-ref name="timer"/>
            <!-- 配置返回页面 -->
            <result>success.jsp</result>
        </action>
 </package>
</struts>
```

在 TestAction 对象的配置中配置了一个拦截器对象 timer，作用是输出 TestAction 执行的时间。

技巧　如果需要查看一个 Action 对象执行所需的时间，可以为其配置 timer 拦截器。它是 Struts 2 的内置拦截器，不需要创建及编写，直接配置即可。

Step 4　创建程序的首页页面 index.jsp 及 TestAction 返回页面 success.jsp，由于实例测试 timer 拦截器的使用，所以没有过多的设置。

部署项目并访问 TestAction 对象，在访问后可以看到 TestAction 执行所占用的时间，如图 15.18 所示。

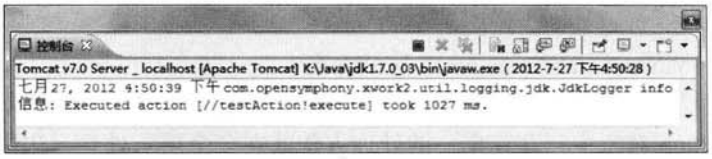


图 15.18　TestAction 执行所占用的时间

说明　访问 TestAction 对象后可看到 TestAction 对象的执行时间大于 1 秒，原因是在第 1 次访问 TestAction 时需要执行一些初始化操作，在以后的访问中即可看到执行时间变为 1 秒（1 000 ms）。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

15.8 数据验证机制

Struts 2 的数据校验机制有两种方式，即通过配置文件和通过重写 ActionSupport 类的 validate 方法。

15.8.1 手动验证

在 Struts 2 的 API 中，ActionSupport 类对 Validateable 接口进行了实现，但对 validate() 方法的实现却是一个空实现，通常情况下，我们所创建的 Action 对象，都是通过继承 ActionSupport 类进行创建的。所以在继承该类的情况下，如果通过 validate() 方法验证数据的有效性，直接重写 validate() 方法即可，如图 15.19 所示。其中 MyAction 类是一个自定义的 Action 对象。

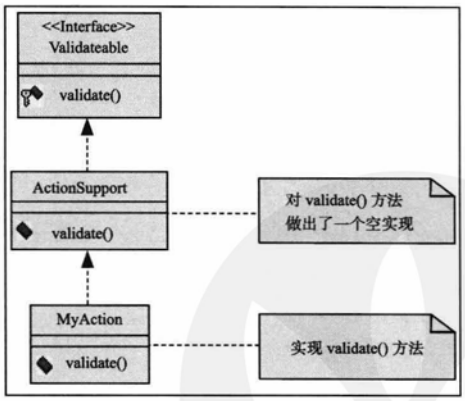


图 15.19 Validateable 与 ActionSupport

使用 validate() 方法可以验证用户请求的多个 Action 方法，并且验证逻辑相同。如果在一个 Action 类中编写了多个请求处理方法，而此 Action 重写了 validate() 方法，那么默认执行每一个请求方法的过程中都会经过 validate() 方法的验证处理。

15.8.2 验证文件的命名规则

使用 Struts 2 验证框架，验证文件名需要遵循一定的命名规则，必须为“ActionName-validation.xml”或“ActionName-AliasName-validation.xml”形式。其中 Action Name 是 Action 对象的名称，AliasName 为 Action 配置中的名称，即 struts.xml 配置文件中 Action 元素对应的 name 属性名。

（1）以“ActionName-validation.xml”方式命名

在这种命名方式中，数据的验证会作用于整个 Action 对象，并验证该对象的请求业务处理方法。如果 Action 对象中只存在单一的处理方法或在多个请求处理方法中验证处理的规则相同，则可以应用此种命名方式。

（2）以“ActionName-AliasName-validation.xml”方式命名

以“ActionName-AliasName-validation.xml”方式命名更加灵活，如果一个 Action 对象中包含多个请求处理方法，而又没有必要验证每一个方法。即只需要处理 Action 对象中的特定方法，则可使用此种命名方式。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

15.8.3　验证文件的编写风格

在 Struts 2 框架中使用数据验证框架，其验证文件的编写有如下两种风格。

Step 1) 字段验证器编写风格

字段验证器编写风格是指在验证过程中主要针对字段进行验证，此种方式是在验证文件根元素 <validators> 下使用 <field-validator> 元素编写验证规则的方式，如：

```
<validators>
 <!-- 验证用户名 -->
 <field name="username">
      <field-validator type="requiredstring">
          <message> 请输入用户名 </message>
      </field-validator>
 </field>
 <!-- 验证密码 -->
 <field name="password">
      <field-validator type="requiredstring">
          <message> 请输入密码 </message>
      </field-validator>
 </field>
</validators>
```

上述代码的作用是判断用户名与密码字段是否输入字符串值。

说明　如果在 xml 文件中使用中文，需要将其字符编码设置为支持中文编码的字符集，如 encoding="UTF-8"。

Step 2) 非字段验证器编写风格

非字段验证器编写风格指在验证过程中既可以针对字段验证，也可以针对普通数据验证。此种方式是在验证文件根元素 <validators> 下使用 <field-validator> 元素编写验证规则的方式，如：

```
<validators>
 <validator type="requiredstring">
       <!-- 验证用户名字段 -->
       <param name="fieldName">password</param>
         !-- 验证密码字段 -->
       <param name="fieldName">username</param>
       <message> 请输入内容 </message>
 </validator>
</validators>
```

上述代码的作用是判断用户名与密码字段是否输入了字符串值。

注意　如果使用非字段验证器编写风格编写验证文件，需要使用 <param> 标签传递字段参数，其参数名为 fieldName，值为字段的名称。

使用第 1 种风格编写的验证文件能够对任何一个字段返回一个明确的验证消息，而使用第 2 种编写验证文件，则不能够对任何一个字段返回一个明确的验证消息，因为这是将多个字段设置在一起。

说明　虽然使用非字段验证器编写风格也能够验证字段，但没有字段验证器编写风格的针对性强，所以验证字段时通常使用字段验证器编写风格。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

15.9 上机实践

15.9.1 应用 Struts 2 处理表单数据

➤➤➤ 题目描述

在应用 Struts 2 时，经常需要处理表单数据。本上机实践尝试创建一个 Java Web 项目并编写一个 Action 对象，处理表单提交的数据，以模拟实现对指定用户的问候，运行结果如图 15.20、图 15.21 所示。



图 15.20 输入用户名页面



图 15.21 提交后的页面

➤➤➤ 技术指导

自定义一个类，即一个 Action 对象，主要用于处理表单提交的信息。如果表单提交的用户名无效，则返回值为 ERROR，程序返回错误页面；否则返回值为 SUCCESS，程序返回成功页面。

Struts 2 为 Action 对象中的属性自动赋值，其内部实现是通过调用 Action 对象中属性提供的 setter 方法赋值。如果没用为属性提供该方法，那么 Struts 2 不能成功地将表单中的数据注入到 Action 的属性中。

15.9.2 Struts 2 标签下的用户注册

➤➤➤ 题目描述

Struts 2 提供的表单标签可以在页面中生成 HTML 表单及表单元素。本上机实践尝试编写一个程序，要求通过 Struts 2 框架提供的表单标签编写用户注册表单，将用户的注册信息输出到 JSP 页面中，运行结果如图 15.22、图 15.23 所示。



图 15.22 用户注册页面

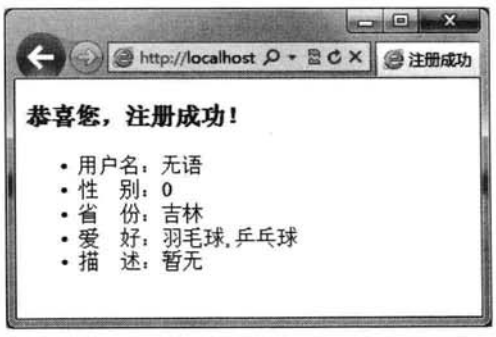


图 15.23 显示注册结果页面

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

➤➤➤ 技术指导

在应用表单时，如果使用 Struts 2 标签生成表单元素，可以使用标签中的 label 属性来定义表单元素前的标签文字。

如果表单中的元素属于一个必填的元素，那么可以使用 required 属性标记，效果为在元素的边缘显示一个字符“*”。另外，通过设置 requiredposition，也可以控制这个字符“*”的位置。

由于表单中的“爱好”对应的表单元素是多选框元素，所以其类型是一个数组对象，在自定义类中用户信息的对应属性是一个字符串数组，本实例将其定义为字符串数组变量。

15.9.3　通过 Struts 2 实现简单投票器

➤➤➤ 题目描述

本上机实践尝试编写一个简单的投票器，要求在投票页面中，使用 Struts 2 的标签显示要投票的候选人及输入选择的候选人文本框，单击“提交”按钮，显示投票人的投票结果，运行结果如图 15.24 和图 15.25 所示。

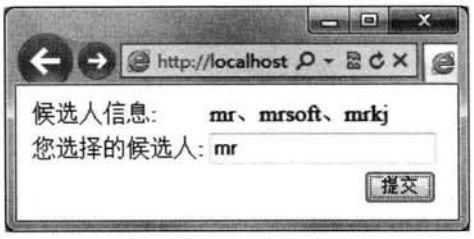


图 15.24　投票页面



图 15.25　显示投票人的投票结果

➤➤➤ 技术指导

在实现本实例时，首先创建投票页面 index.jsp，在该页面中，应用 Struts 2 的标签显示候选人信息、选择的候选人和提交按钮；然后编写继承了 ActionSupport 的自定义 Action 类，在该类中，定义 name 属性，并为该属性创建对应的 getter 和 setter 方法，这一点很重要，在请求该 Action 时，Struts 2 会自动将表单的值注入到该属性中；最后编写显示投票结果的页面 result.jsp，在该页面中显示投票人的投票结果。

15.10　小结

本章主要介绍了 Struts 2 框架的基础知识及其高级技术，读者应重点掌握 Struts 2 的原理、处理流程及 Struts 2 标签等常用技术。

15.11　习题

选择题

1．Action 中的默认方法是（　　）。

A. doPost()　　B. execute()　　C. doGet()

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2. Struts 2 配置文件 struts.xml 中 result 标记的默认类型是（　　）。

A. dispatcher　　B. redirect　　C. chain

3. Struts 2 中定义的 Action 类都要直接或者间接地实现（　　）接口。

A. Filter　　B. Servlet　　C. Action

4. 在 JSP 中引入 struts 2 标签库的指令是（　　）。

A. page　　B. include　　C. taglib

5. Struts 2 配置文件 struts.xml 默认位置在（　　）路径下。

A. /WEB-INF/　　B. /WEB-INF/classes/　　C. /WEB-INF/lib/

填空题

1. 配置 Struts 2 提供的过滤器是在_____配置文件中进行配置。
2. 在 struts.xml 中声明名称空间用到的属性是_____。
3. 在 Struts 2 中的零配置实现就是在 Action 类中使用_____定义 Action 的资源。
4. 在 struts.xml 中实现 action 链要配置的 type 属性是_____。
5. 在 Struts 2 中获取数据值，并将数据值直接输出到页面之中的标签是_____。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 16 章　数据库持久化利器——Hibernate 技术

视频讲解：85 分钟

作为一个优秀的持久层框架，Hibernate 充分体现了 ORM 的设计理念，提供了高效的对象到关系型数据库的持久化服务。它将持久化服务从软件业务层中完全提取出来，让业务逻辑的处理更加简单。Hibernate 的使用使程序之间的各种业务并非紧密耦合，从而更加有利于高效的开发与维护开发。本章将详细介绍 Hibernate 的知识。

通过学习本章，读者可以达到以下学习目的：

- 了解 ORM 原理；
- 掌握如何获取并配置 Hibernate；
- 掌握如何使用 Hibernate 进行数据持久化；
- 掌握使用 Hibernate 的缓存；
- 掌握如何配置实体关联关系映射；
- 掌握如何使用 Hibernate 查询语言。

16.1　初识Hibernate

16.1.1　ORM 原理

目前面向对象思想是软件开发的基本思想，关系数据库又是应用系统中必不可少的一环。但是面向对象思想是从软件工程的基本原则发展而来，而关系数据库却是基于数学理论，二者的区别巨大。为了解决这个问题，ORM（Object Relational Mapping，对象到关系的映射）便应运而生。

ORM 的作用是在关系数据库和对象之间做一个自动映射，将数据库中的数据表映射成为对象，即持久化类。以对象的形式操作数据库，减少应用开发过程中数据持久化的编程任务。可以把 ORM 理解为关系型数据和对象的一个纽带，开发人员只需关注纽带一端映射的对象即可。ORM 原理如图 16.1 所示。

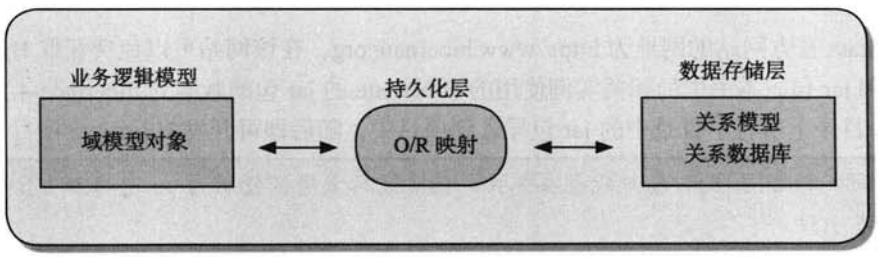


图 16.1　ORM 原理

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Hibernate 是众多 ORM 工具中的佼佼者，相对于 iBATIS（一个基于 Java 的持久层框架），它是全自动的关系 / 对象的解决方案。Hibernate 通过持久化类（*.java）、映射文件（*.hbm.xml）和配置文件（*.cfg.xml）操作关系型数据库，使开发人员不必再与复杂的 SQL 语句打交道。

16.1.2 Hibernate 简介

Hibernate 是 Java Web 开发中最常用的一个持久层框架，它充分体现了 ORM 的设计理念，它能够将持久化服务从软件业务层中完全提取出来，使业务逻辑的处理更加简单。图 16.2 所示为 Hibernate 的体系结构概要。

从图 16.2 中可以清楚地看出，Hibernate 是通过数据库和配置信息执行数据持久化服务和持久化对象的，它封装了数据库的访问细节，通过配置的属性文件这条纽带连接关系型数据库和程序中的实体类。

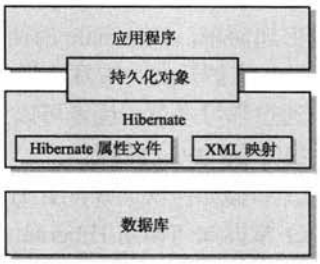


图 16.2 Hibernate 的体系结构概要

在 Hibernate 中有如下 3 个重要的类。

（1）配置类（Configuration）

配置类主要负责管理 Hibernate 的配置信息及启动 Hibernate，在 Hibernate 运行时该类会读取一些底层实现的基本信息，其中包括数据库 URL、数据库用户名、数据库用户密码、数据库驱动类和数据库适配器（dialect）等。

（2）会话工厂类（SessionFactory）

会话工厂类是生成 Session 的工厂，保存当前数据库中所有的映射关系。会话工厂类可能只有一个可选的二级数据缓存，并且它是线程安全的。该类是一个重量级对象，其初始创建过程会耗费大量的系统资源。

（3）会话类（Session）

会话类是 Hibernate 中数据库持久化操作的核心，负责 Hibernate 所有的持久化操作，通过它可以实现数据库基本的增、删、改和查操作。该类不是线程安全的，应注意不要多个线程共享一个 Session。

16.2 Hibernate入门

16.2.1 获取 Hibernate

Hibernate 官方网站的网址为 http://www.hibernate.org，在该网站可以免费获取 Hibernate 的帮助文档和 jar 包。本书中的相关实例使用的 Hibernate 的 jar 包的版本为 hibernate-4.1.4。

将 lib 目录下所有子目录中的 jar 包导入到项目中，随后即可开发 Hibernate 项目。

注意 不同版本之间的 jar 包可能存在差异，读者应尽量保证使用的 jar 包与本书实例所用的版本一致。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

16.2.2　Hibernate 配置文件

在 Hibernate 4.1.4 版本中，默认从 hibernate.properties 文件中加载数据库的配置信息，该文件默认存放于项目的 classpath 根目录下。例如，要连接 MySQL 数据库，需要编写下面的 hibernate.properties 文件。

```
# 数据库驱动
hibernate.connection.driver_class = com.mysql.jdbc.Driver
# 数据库连接的 URL
hibernate.connection.url = jdbc:mysql://localhost:3306/db_database16
# 用户名
hibernate.connection.username = root
# 密码
hibernate.connection.password = 111
# 是否显示 SQL 语句
hibernate.show_sql=true
#Hibernate 方言
hibernate.dialect = org.hibernate.dialect.MySQLDialect
```

另外，Hibernate 还可以通过读取 XML 配置文件 hibernate.cfg.xml 加载数据库的配置信息，该配置文件默认放于项目的 classpath 根目录下。下面我们来看一下连接 MySQL 数据库所用的 XML 配置文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
 <hibernate-configuration>
   <session-factory>
          <!-- 数据库驱动 -->
          <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
          <!-- 数据库连接的 URL -->
          <property name="connection.url">jdbc:mysql://localhost:3306/db_
          database16</property>
          <!-- 数据库连接用户名 -->
          <property name="connection.username">root</property>
          <!-- 数据库连接密码 -->
          <property name="connection.password">111</property>
          <!-- Hibernate 方言 -->
          <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
          <!-- 打印 SQL 语句 -->
          <property name="show_sql">true</property>
          <!-- 映射文件  -->
          <mapping resource="com/mr/employee/Employee.hbm.xml"/>
          <mapping resource="com/mr/user/User.hbm.xml"/>
          </session-factory>
 </hibernate-configuration>
```

从配置文件中可以看出配置信息包括整个数据库的信息，如数据库的驱动、URL 地址、用户名、密码和 Hibernate 使用的方言，还需要管理程序中各个数据表的映射文件。

Hibernate 提供的常用属性如表 16.1 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 16.1 Hibernate 提供的属性表

属 性	说 明
connection.driver_class	连接数据库的驱动
connection.url	连接数据库的 URL 地址
connection.username	连接数据库用户名
connection.password	连接数据库密码
dialect	连接数据库使用的方言
show_sql	是否在控制台打印 SQL 语句
format_sql	是否格式化 SQL 语句
hbm2ddl.auto	是否自动生成数据表

在程序开发过程中，一般会将 show_sql 属性设置为 true，以便在控制台打印自动生成的 SQL 语句，方便程序的调试。

以上只是 Hibernate 配置的一部分，还包含其他配置属性，如还可以配置表的自动生成和 Hibernate 的数据连接池等。

16.2.3 编写持久化类

在 Hibernate 中，持久化类是其操作的对象，即通过对象 - 关系映射（ORM）后数据表所映射的实体类描述数据表的结构信息，在持久化类中的属性应该与数据表中的字段相匹配。例如，对于一个包括 id、name 和 password 三个字段的数据表，我们可以创建下面的持久化类。

```
public class User {
    private Integer id;                                    //用户 ID
    private String name;                                   //用户名
    private String password;                               //用户密码
    //默认的构造方法
    public User(){
    }
    public Integer getId() {
            return id;
    }
    public void setId(Integer id) {
            this.id = id;
    }
... //此处省略了其他属性的 setter 和 getter 方法
}
```

User 类作为一个简单的持久化类，符合基本的 JavaBean 编码规范，即 POJO（Plain Old Java Object）编程模型。持久化类中的每个属性都有相应的 setter 和 getter 方法，不依赖于任何接口和继承任何类。

说明 POJO 编程模型指普通的 JavaBean，通常有一些参数作为对象的属性。然后为每个属性定义 getter 和 setter 方法作为访问接口，它被大量应用于表现现实中的对象。

Hibernate 中的持久化类有如下 4 条编程规则。

（1）实现一个默认的构造函数

所有的持久化类中都必须含有一个默认的无参数构造方法（User 类中含有这种方法），以

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

便 Hibernate 通过 Constructor.newInstance() 实例化持久化类。

（2）提供一个标识属性（可选）

标识属性一般映射的是数据表中的主键字段，如 User 中的属性 id，建议在持久化类中添加一致的标识属性。

（3）使用非 final 类（可选）

如果使用了 final 类，Hibernate 不能使用代理来延迟关联加载，这会影响性能优化的选择。

（4）为属性声明访问器（可选）

持久化类的属性不能声明为 public，最好以 private 的 set() 和 get() 方法持久化属性。

16.2.4　Hibernate 映射

Hibernate 的核心是对象关系映射，对象和关系型数据库之间的映射通常用 XML 文档来实现。这个映射文档被设计为易读，并且可以手工修改。映射文件的命名规则为 *.hbm.xml。以 User 的持久化类的映射文件为例，其代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<!-- 对持久化类 User 的映射配置 -->
<hibernate-mapping>
   <class name="com.mr.User" table="tb_user">
          <!-- 持久化类的唯一性标识 -->
          <id name="id" column="id" type="int">
                <generator class="native"/>
          </id>
          <property name="name" type="string" not-null="true" length="50">
                <column name="name"/>
          </property>
          <property name="password" type="string" not-null="true" length="50">
                <column name="password"/>
          </property>
   </class>
</hibernate-mapping>
```

说明　映射语言以 Java 为中心，所以映射文档按照持久化类的定义，而不是数据表的定义创建。

（1）<DOCTYPE> 元素

在所有的 Hibernate 映射文件中都需要定义 <DOCTYPE> 元素来获取 DTD 文件。

（2）<hibernate-mapping> 元素

<hibernate-mapping> 元素是映射文件中其他元素的根元素，其中包含一些可选的属性，如 schema 属性指定该文件映射表所在数据库的 schema 名称，package 属性指定一个包的前缀。如果在 <class> 元素中没有指定全限定的类名，则使用 package 属性定义的包前缀作为包名。

（3）<class> 元素

<class> 元素主要用于指定持久化类和映射的数据表名，name 属性需要指定持久化类的全限定的类名（如“com.mr.User”）；table 属性是持久化类所映射的数据表名。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

<class> 元素中包含了一个 <id> 元素和多个 <property> 元素，前者用于持久化类的唯一标识与数据表的主键字段的映射，在其中通过 <generator> 元素定义主键的生成策略；后者用于持久化类的其他属性和数据表中非主键字段的映射，其主要属性如表 16.2 所示。

表 16.2 <property> 元素的常用属性

属性名称	说　明
name	持久化类属性的名称，以小写字母开头
column	数据库字段名
type	数据库的字段类型
length	数据库字段定义的长度
not-null	该数据库字段是否可以为空，该属性为布尔变量
unique	该数据库字段是否唯一，该属性为布尔变量
lazy	是否延迟抓取，该属性为布尔变量

注意 如果在映射文件中没有配置 column 和 type 属性，Hibernate 将默认使用持久化类中的属性名称和属性类型匹配数据表中的字段。

16.2.5 Hibernate 主键策略

<id> 元素的子元素 <generator> 元素是一个 Java 类名，用来为持久化类的实例生成唯一的标识以映射数据库中的主键字段。在配置文件中通过设置 <generator> 元素的属性设置 Hibernate 的主键生成策略，常用属性如表 16.3 所示。

表 16.3 Hibernate 主键生成策略的常用属性

属性名称	说　明
increment	用于为 long、short 或者 int 类型生成唯一标识，在集群下不要使用该属性
identity	由底层数据库生成主键，前提是底层数据库支持自增字段类型
sequence	根据底层数据库的序列生成主键，前提是底层数据库支持序列
hilo	根据高 / 低算法生成，把特定表的字段作为高位值来源，在默认的情况下选用 hibernate_unique_key 表的 next_hi 字段
native	根据底层数据库对自动生成标识符的支持能力选择 identity、sequence 或 hilo
assigned	由程序负责主键的生成，此时持久化类的唯一标识不能声明为 private 类型
select	通过数据库触发器生成主键
foreign	使用另一个相关联的对象的标识符，通常和 <one-to-one> 一起使用

16.3 使用Hibernate进行数据持久化

持久化操作是 Hibernate 的核心，本节讲解如何创建线程安全的 Hibernate 初始化类，并利用 Hibernate 的 Session 对象实现基本的数据库增、删、改和查的操作，然后说明 Hibernate 的延迟加载策略，以优化系统的性能。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

16.3.1　Hibernate 实例状态

Hibernate 的实例状态分为如下 3 种。

（1）瞬时状态（Transient）

实体对象通过 Java 中的 new 关键字开辟内存空间创建 Java 对象，但是它并没有纳入 Hibernate Session 的管理。如果没有变量对它引用，则其将被 JVM（垃圾回收器）回收。瞬时状态的对象在内存中是孤立存在的，与数据库中的数据是无任何关联，仅仅是一个信息携带的载体。

假如一个瞬时状态对象被持久化状态对象引用，它也会自动变为持久化状态对象。

（2）持久化状态（Persistent）

持久化状态对象存在与数据库中的数据关联，总是与会话状态（Session）和事务（Transaction）关联在一起，当持久化状态对象发生改动时并不会立即执行数据库操作。只有当事务结束时，才会更新数据库，以便保证 Hibernate 的持久化对象和数据库操作的同步性。当持久化状态对象变为脱管状态对象时，它将不在 Hibernate 持久层的管理范围之内。

（3）脱管状态（Detached）

当持久化状态对象的 Session 关闭之后，这个对象就从持久化状态的对象变为脱管状态的对象。脱管状态的对象仍然存在与数据库中的数据关联，只是不在 Hibernate 的 Session 管理范围之内。如果将脱管状态的对象重新关联某个新的 Session，则将变回持久化状态对象。

Hibernate 中的 3 种实例状态的关系如图 16.3 所示。

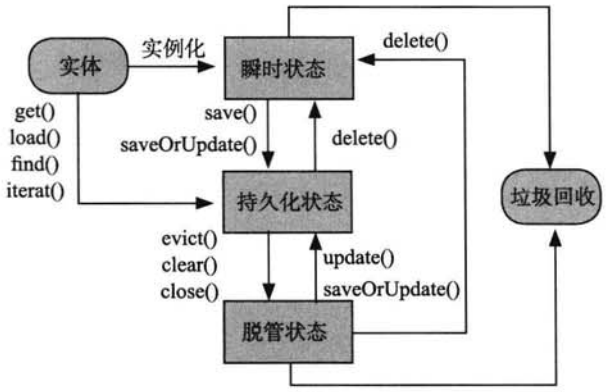


图 16.3　Hibernate 中的 3 种实例状态的关系

16.3.2　Hibernate 初始化类

Session 对象通过 SessionFactory 对象获取，可以通过 Configuration 对象创建 SessionFactory，关键代码如下：

```
Configuration cfg = new Configuration().configure();        //加载 Hibernate 配
                                                              置文件
factory =cfg.buildSessionFactory(new ServiceRegistryBuilder().
buildServiceRegistry());                                    // 实例化 SessionFactory
```

Configuration 对象会加载 Hibernate 的基本配置信息，如果没有在 configure() 方法中指定加载配置 XML 文档的路径信息，Configuration 对象会默认加载项目 classpath 根目录下的

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

hibernate.cfg.xml 文件。

说明 在实例化 SessionFactory 时，如果是 Hibernate 3，那么就不需要为 buildSessionFactory() 方法指定参数，也就是说，可以使用下面的代码来实例化 SessionFactory。

```
factory =cfg.buildSessionFactory();        // 在 Hibernate 3 中实例化 SessionFactory
```

【例 16.1】 创建 Hibernate 初始化类。

实例位置：光盘 \MR\ 源码 \ 第 16 章 \16.1

```
public class HibernateUtil {
   private static final ThreadLocal<Session> threadLocal = new ThreadLocal
<Session>();
private static SessionFactory sessionFactory = null;    //SessionFactory 对象
// 静态块
static {
          try {
                Configuration cfg = new Configuration().configure();
                                                        // 加载 Hibernate 配置文件
                sessionFactory = cfg.buildSessionFactory(new
                ServiceRegistryBuilder().buildServiceRegistry());
          } catch (Exception e) {
                System.err.println("创建会话工厂失败");
                e.printStackTrace();
          }
    }
   /**
     * 获取 Session
     *  @return Session
     *  @throws HibernateException
     */
    public static Session getSession() throws HibernateException {
        Session session = (Session) threadLocal.get();
          if (session == null || !session.isOpen()) {
                if (sessionFactory == null) {
                      rebuildSessionFactory();
                }
                session = (sessionFactory != null) ? sessionFactory.
                openSession(): null;
                threadLocal.set(session);
          }
        return session;
    }
    /**
     * 重建会话工厂
     */
    public static void rebuildSessionFactory() {
          try {
                Configuration cfg = new Configuration().configure();
                                                        // 加载 Hibernate 配置文件
                sessionFactory = cfg.buildSessionFactory(new
                ServiceRegistryBuilder().buildServiceRegistry());
          } catch (Exception e) {
                System.err.println("创建会话工厂失败");
                e.printStackTrace();
          }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    }
    /**
     * 获取 SessionFactory 对象
     * @return SessionFactory 对象
     */
    public static SessionFactory getSessionFactory() {
            return sessionFactory;
    }
    /**
      * 关闭 Session
      *  @throws HibernateException
      */
     public static void closeSession() throws HibernateException {
         Session session = (Session) threadLocal.get();
         threadLocal.set(null);
         if (session != null) {
             session.close();                              //关闭 Session
         }
     }
}
```

在上面的代码中，由于 SessionFactory 是重量级的对象，其创建需要耗费大量的系统资源。所以将 SessionFactory 的创建放在静态块中，程序运行过程中只创建一次。

说明 由于 SessionFactory 是线程安全的，但是 Session 不是，所以让多个线程共享一个 Session 对象可能会引起数据的冲突。为了保证 Session 的线程安全，引入了 ThreadLocal 对象，避免多个线程之间的数据共享。

16.3.3　保存数据

Hibernate 对 JDBC 的操作进行了轻量级的封装，使开发人员可以利用 Session 对象以面向对象的思想实现对关系型数据库的操作。Hibernate 的数据持久化过程如图 16.4 所示。

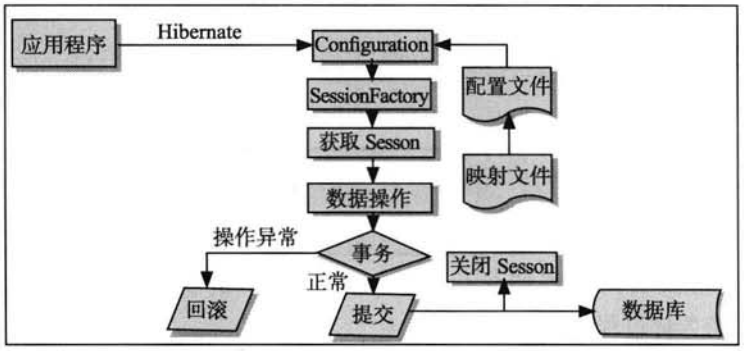


图 16.4　Hibernate 的数据持久化过程

接下来的讲解以商品的基本信息为例，执行数据库的增、删、改和查操作，首先构造商品的持久化类 Product.java，其关键代码如下：

```
private Integer id;                    //唯一性标识
private String name;                   //产品名称
private Double price;                  //产品价格
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
private String factory;                    // 生产商
private String remark;                     // 备注
…                                          // 省略了各属性的 setter 和 getter 方法
```

在执行添加操作时需要 Session 对象的 save() 方法，其入口参数为程序中的持久化类。

【例 16.2】 向数据库的产品信息表中添加产品信息。

实例位置：光盘 \MR\ 源码 \ 第 16 章 \16.2

创建一个 Servlet，在其 doPost() 方法中编写向数据表中添加产品信息的代码，关键代码如下：

```
protected void doPost(HttpServletRequest request,HttpServletResponse response)
throws ServletException, IOException {
        Session session = null;                          // 声明 Session 对象
        response.setContentType("text/html;charset=utf-8");
                                                         // 设置内容类型，防止中文乱码
        Product product = new Product();                 // 实例化持久化类
        // 为持久化类属性赋值
        product.setName("Java Web 编程宝典 ");           // 设置产品名称
        product.setPrice(79.00);                         // 设置产品价格
        product.setFactory(" 明日科技 ");                // 设置生产商
        product.setRemark(" 无 ");                       // 设置备注
        // Hibernate 的持久化操作
        try {
            session = HibernateUtil.getSession();        // 获取 Session
            session.beginTransaction();                  // 开启事务
            session.save(product);                       // 执行数据库添加操作
            session.getTransaction().commit();           // 事务提交
            PrintWriter out= response.getWriter();
            out.println("<script>alert(' 数据添加成功！ ');</script>");
        } catch (Exception e) {
            session.getTransaction().rollback(); // 事务回滚
            System.out.println(" 数据添加失败 ");
            e.printStackTrace();
        } finally {
            HibernateUtil.closeSession();                // 关闭 Session 对象
        }
    }
```

说明 事务可以保证数据操作的一致性，在本实例中添加事务没有意义，只是为了演示 Hibernate 的基本事务配置。

读者可以根据该实例分析持久化对象 product 的实例状态和流程，这样更利于理解 Hibernate 的数据持久化过程。

说明 持久化对象 product 在创建之后是瞬时状态（Transient），在 Session 执行 save() 方法之后变为持久化状态（Persistent），但是这时数据操作并未提交给数据库。在事务执行 commit() 方法之后才完成数据库的添加操作，此时的持久化对象 product 成为脏（dirty）对象。Session 关闭之后，状态变为脱管状态（Detached），并最后被 JVM 所回收。

程序运行后，在 tb_product 表中添加的信息如图 16.5 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

id	name	price	factory	remark
1	Java Web编程宝典	79	明日科技	无

图 16.5　在 tb_product 表中添加的信息

16.3.4　查询数据

Session 对象提供了以下两种加载对象的方法。

（1）get() 方法

如果不确定数据库中是否有匹配的记录存在，即可使用 get() 方法加载对象，因为它会立刻访问数据库。如果数据库中没有匹配记录存在，则会返回 null。

【例 16.3】　利用 get() 方法加载 Product 对象。

实例位置：光盘 \MR\ 源码 \ 第 16 章 \16.3

创建一个 Servlet，在其 doPost() 方法中编写利用 get() 方法加载产品对象的代码，关键代码如下：

```
response.setContentType("text/html;charset=utf-8");
                                                    //设置内容类型，防止中文乱码
Session session = null;                             //声明 Session 对象
try {
                                                    //Hibernate 的持久化操作
      session = HibernateUtil.getSession(); //获取 Session
      Product product = (Product) session.get(Product.class, new
      Integer("1"));                                //装载对象
      …    //此处省略了输出获取结果的代码
} catch (Exception e) {
      System.out.println("对象装载失败");
      e.printStackTrace();
} finally{
      HibernateUtil.closeSession();                 //关闭 Session
}
```

运行本实例，在 JSP 页面中将显示如图 16.6 所示的获取结果。

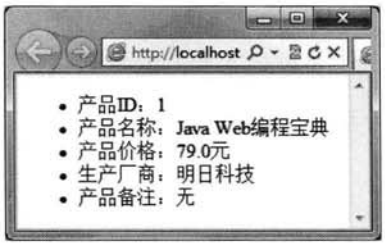


图 16.6　通过 get() 方法获取产品对象

（2）load() 方法

load() 方法返回对象的代理，只有在返回对象被调用时 Hibernate 才会发出 SQL 语句查询对象。

【例 16.4】　利用 load() 方法加载 Product 对象。

实例位置：光盘 \MR\ 源码 \ 第 16 章 \16.4

使用 load() 方法加载产品对象与例 16.3 中使用 get() 方法加载产品对象类似，也需要创

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

建一个 Servlet，在其 doPost() 方法中编写利用 load() 方法加载产品对象的代码，关键代码如下：

```
session = HibernateInitialize.getSession();                    // 获取 Session
Product product = (Product) session.load(Product.class, new Integer("1"));
                                                               // 加载对象
```

另外 load() 方法还可以加载到指定的对象实例上，代码如下：

```
session = HibernateInitialize.getSession();                    // 获取 Session
Product product = new Product();                               // 实例化对象
session.load(product, new Integer("1"));                       // 加载对象
```

上面两种方法的运行结果相同，程序运行后，都将显示如图 16.6 所示的运行结果。

16.3.5 删除数据

在 Session 对象中需要使用 delete() 方法删除数据，但是只有对象在持久化状态时才能执行该方法，所以在删除数据之前需要将对象的状态转换为持久化状态。

【例 16.5】 利用 delete() 方法删除指定的产品信息。

实例位置：光盘 \MR\ 源码 \ 第 16 章 \16.5

创建一个 Servlet，在其 doPost() 方法中编写利用 delete() 方法删除指定的产品信息的代码，关键代码如下：

```
try {
   //Hibernate 的持久化操作
   session = HibernateUtil.getSession();                       // 获取 Session
   session.beginTransaction();                                 // 开启事务
   Product product = (Product) session.get(Product.class, new Integer("1"));
                                                               // 加载对象
   session.delete(product);                                    // 删除持久化对象
   session.flush();                                            // 强制刷新提交
   session.getTransaction().commit();                          // 事务提交
   System.out.println(" 对象删除成功！ ");
} catch (Exception e) {
   session.getTransaction().rollback();                        // 事务回滚
   System.out.println(" 对象装载失败 ");
   e.printStackTrace();
} finally{
   HibernateUtil.closeSession();                               // 关闭 Session
}
```

程序运行后控制台输出的信息如图 16.7 所示。

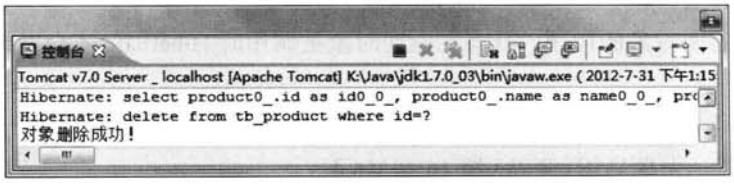


图 16.7 控制台输出的信息

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

16.3.6　修改数据

在 Hibernate 的 Session 管理中，如果程序修改了持久化状态的对象，当 Session 刷出时，Hibernate 会对实例执行持久化操作，利用该特性即可修改商品信息。

说明　Session 的刷出（flush）过程指 Session 执行一些必需的 SQL 语句来把内存中对象的状态同步到 JDBC 中。刷出会在某些查询之前、在事务提交时或者在程序中直接调用 Session.flush() 时执行。

【例 16.6】 修改指定的产品信息。

实例位置：光盘\MR\源码\第 16 章\16.6

创建一个 Servlet，在其 doPost() 方法中编写修改指定产品信息的代码，关键代码如下：

```
Session session = null;                                    // 声明 Session 对象
      try {
          //Hibernate 的持久化操作
          session = HibernateUtil.getSession();            // 获取 Session
          session.beginTransaction();                      // 开启事务
          Product product = (Product) session.get(Product.class, new
          Integer("1"));                                   // 加载对象
          product.setName("Java Web 编程词典 ");           // 修改商品名称
          product.setRemark(" 明日科技出品 ");             // 修改备注信息
          session.flush();                                 // 强制刷新提交
          session.getTransaction().commit();               // 事务提交
          System.out.println(" 对象修改成功！ ");
      } catch (Exception e) {
          session.getTransaction().rollback();             // 事务回滚
          System.out.println(" 对象装载失败 ");
          e.printStackTrace();
      } finally{
          HibernateUtil.closeSession();                    // 关闭 Session
      }
```

程序运行前，数据库中的信息如图 16.8 所示。程序运行后，数据库中的信息如图 16.9 所示。

id	name	price	factory	remark
1	Java Web编程宝典	79	明日科技	无

图 16.8　程序运行前数据库中的信息

id	name	price	factory	remark
1	Java Web编程词典	79	明日科技	明日科技出品

图 16.9　程序运行后数据库中的信息

16.3.7　延迟加载

在使用 load() 方法加载持久化对象时，返回的是一个未初始化的代理（代理无须从数据库中提取数据对象的数据），直到调用代理的某个方法时 Hibernate 才会访问数据库。在非延迟加载过程中，Hibernate 会直接访问数据库，并不会使用代理对象。

延迟加载策略的原理如图 16.10 所示。

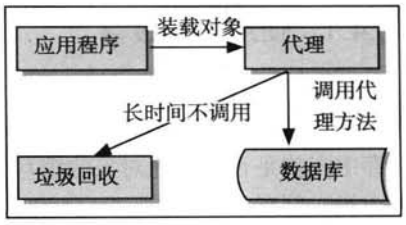


图 16.10　延迟加载策略的原理

当加载的对象长时间没有调用时，就会被垃圾回收器回收，在程序中合理地使用延迟加载策略可优化系统的性能。采用延迟加载可以使 Hibernate 节省系统的

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

内存空间，否则每加载一个持久化对象就需要将其关联的数据信息加载到内存中，这将为系统增加不必要的开销。

在 Hibernate 中可以通过使用一些采用延迟加载策略封装的方法实现延迟加载的功能，如 load() 方法，同时还可以通过设置映射文件中的 <property> 元素中的 lazy 属性实现该功能。以产品信息的 XML 文档配置为例进行介绍，关键代码如下：

```
<!-- 产品信息字段配置信息 -->
<hibernate-mapping>
   <class name="com.wgh.product.Product" table="tb_product">
          <!-- id值 -->
          <id name="id" column="id" type="int">
          <generator class="native"/>
   </id>
   <!-- 产品名称 -->
   <property name="name" type="string" length="45" lazy="true">
          <column name="name"/>
   </property>
   …
   </class>
</hibernate-mapping>
```

通过该方式产品名称属性被设置为延迟加载。

16.4 使用Hibernate的缓存

缓存是数据库数据在内存中的临时容器，是数据库与应用程序的中间件，如图 16.11 所示。

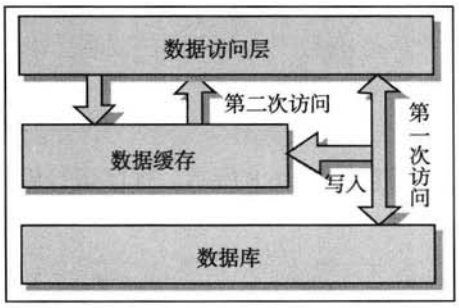


图 16.11 数据缓存

在 Hibernate 中采用了缓存的技术，使其可以更加高效地执行数据持久化操作。Hibernate 数据缓存分为一级缓存（Session Level，也称为“内部缓存”）和二级缓存（SessionFactory Level）。

16.4.1 使用一级缓存

Hibernate 的一级缓存属于 Session 级缓存，所以其生命周期与 Session 相同。

当程序使用 Session 加载持久化对象时，Session 首先会根据加载的数据类和唯一性标识在缓存中查找是否存在此对象的缓存实例。如果存在，将其作为结果返回；否则继续在二级缓存中查找实例对象。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

注意　在 Hibernate 中不同的 Session 之间不能共享一级缓存，即一个 Session 不能访问其他 Session 在一级缓存中的对象缓存实例。

【例 16.7】 在同一 Session 中查询两次产品信息。

实例位置：光盘 \MR\ 源码 \ 第 16 章 \16.7

创建一个 Servlet，在其 doPost() 方法中编写在同一 Session 中查询两次产品信息的代码，关键代码如下：

```
    try {
    //Hibernate 的持久化操作
    session = HibernateUtil.getSession();                    // 获取 Session
    Product product = (Product) session.get(Product.class, new Integer("1"));
                                                             // 加载对象
    System.out.println(" 第一次加载对象 ");
    Product product2 = (Product) session.get(Product.class, new Integer("1"));
                                                             // 加载对象
    System.out.println(" 第二次加载对象 ");
} catch (Exception e) {
    e.printStackTrace();
} finally{
    HibernateInitialize.closeSession();                      // 关闭 Session
}
```

程序运行后控制台输出的信息如图 16.12 所示。

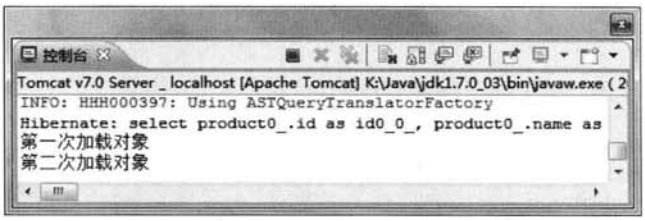


图 16.12　控制台输出的信息

从控制台输出信息中可以看出 Hibernate 只访问了一次数据库，第 2 次对象加载时是从一级缓存中将该对象的缓存实例以结果形式直接返回。

16.4.2　配置和使用二级缓存

Hibernate 的二级缓存由从属于一个 SessionFactory 的所有 Session 对象共享，当程序使用 Session 加载持久化对象时，Session 首先会根据加载的数据类和唯一性标识在缓存中查找是否存在此对象的缓存实例。如果存在，将其作为结果返回；否则继续在二级缓存中查找。如果无匹配对象，Hibernate 将直接访问数据库。

由于 Hibernate 本身并未提供二级缓存的产品化实现，所以需要引入第三方插件实现二级缓存的策略。本节将以 EHCache 作为 Hibernate 默认的二级缓存，讲解 Hibernate 二级缓存的配置及其使用方法。

【例 16.8】 利用二级缓存查询产品信息。

实例位置：光盘 \MR\ 源码 \ 第 16 章 \16.8

首先需要在 Hibernate 配置文件 hibernate.cfg.xml 中配置开启二级缓存，关键代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<hibernate-configuration>
   <session-factory>
          ...
          <!-- 开启二级缓存 -->
          <property name="hibernate.cache.use_second_level_cache">true</
          property>
          <!-- 指定缓存产品提供商 -->
          <property name="hibernate.cache.region.factory_class">
               org.hibernate.cache.ehcache.EhCacheRegionFactory</property>
   </session-factory>
</hibernate-configuration>
```

在持久化类的映射文件中需要指定缓存的同步策略，关键代码如下：

```
<!—产品信息字段配置信息 -->
<hibernate-mapping>
   <class name="com.wgh.product.Product" table="tb_product">
          <!-- 指定的缓存的同步策略 -->
          <cache usage="read-only"/>
          ...
   </class>
</hibernate-mapping>
```

在项目的 classpath 根目录下添加缓存配置文件 ehcache.xml，此文件可以从 Hibernate 的 zip 包下的 etc 目录中找到，其代码如下：

```
<ehcache>
<diskStore path="java.io.tmpdir"/>
<defaultCache
          maxElementsInMemory="10000"
          eternal="false"
          timeToIdleSeconds="120"
          timeToLiveSeconds="120"
          overflowToDisk="true"
   />
</ehcache>
```

在上面的代码中，maxElementsInMemory 属性用于设置缓存中允许保存的最大数据实例数量；eternal 属性用于设置缓存中的数据是否为常量；timeToIdleSeconds 属性用于设置缓存数据钝化时间；timeToLiveSeconds 属性用于设置缓存数据生存时间；overflowToDisk 属性用于设置是否启用磁盘缓存。

创建一个 Servlet，在其 doPost() 方法中编写在同一 SessionFactory 中获取两个 Session 的代码，要求每个 Session 执行一次 get() 方法，关键代码如下：

```
Session session = null;                                    // 声明第 1 个 Session 对象
Session session2 = null;                                   // 声明第 2 个 Session 对象
try {
   //Hibernate 的持久化操作
   session = HibernateUtil.getSession();                   // 获取第 1 个 Session
   session2 = HibernateUtil.getSession();                  // 获取第 2 个 Session
   Product product = (Product) session.get(Product.class, new Integer("1"));
                                                           // 加载对象
   System.out.println(" 第一个 Session 加载对象 ");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        Product product2 = (Product) session2.get(Product.class, new Integer("1"));
                                                                    // 加载对象
        System.out.println("第二个 Session 加载对象 ");
    } catch (Exception e) {
        e.printStackTrace();
    } finally{
        HibernateUtil.closeSession();                               // 关闭 Session
    }
```

程序运行后控制台输出的信息如图 16.13 所示。

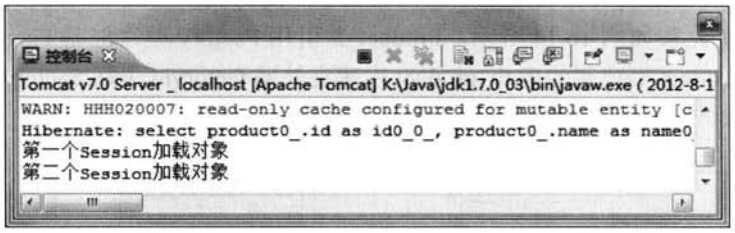


图 16.13　控制台输出的信息

当第 2 个 Session 加载对象时，控制并没有输出 SQL 语句，说明 Hibernate 是从二级缓存中加载的该实例对象。二级缓存是常用于数据更新频率低且系统频繁使用的非关键数据，以防止用户频繁访问数据库而过度消耗系统资源。

16.5　实体关联关系映射

ORM 是 Hibernate 的理论基础，所以映射在 Hibernate 中占有重要的地位。本节介绍在 Hibernate 中如何通过映射将持久化类与数据表关联。

16.5.1　数据模型与领域模型

了解数据模型与领域模型有助于更好地理解实体对象的关联关系映射。

（1）数据模型

数据模型是对数据库特征的抽象，即用户从数据库中看到的模型，例如，一个数据表或者用户从数据表中看到的信息。此模型面向用户和系统，面向用户需要将存储数据完整地展现在用户面前，使用户可以对数据执行增、删、改和查操作；面向系统是通知计算机如何有效地管理数据，主要用于数据库管理系统（DBMS）的实现。

说明　数据模型是数据库管理系统（DBMS）设计实现的一部分，描述的是用户需求在数据结构上的实现，但是缺少实体对象之间的关系描述。

（2）领域模型

领域模型是对现实世界中的对象的可视化表现，又称为“概念模型”、“领域对象模型”或“分析对象模型”。

领域模型主要是特定群体中用于理解和沟通的工具，有效的领域模型能够捕获当前需求语境下的本质抽象和理解领域所需要的信息，并且可以帮助人们理解领域的概念、术语和关系。它是现实世界与计算机之间的一条无形的纽带，也是需求分析设计人员一件强有

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

力的工具。

领域模型也是一个分析模型，是帮助需求分析人员和用户认识现实业务的工具。它描述的是业务中涉及的实体及其之间的关系，是需求分析的产物。领域模型设计期间不用考虑数据的管理问题，只考虑业务描述中涉及的实体与实体之间的关系。

16.5.2 理解并配置多对一单向关联

关联是类（类的实例）之间的关系，表示有意义和值得关注的连接。

单向多对一的映射实现比较简单。以产品和生产商为例。实现单向的多对一映射，两个持久化类的依赖关系如图 16.14 所示。Product 类引用 Factory 类，但是 Factory 类没有引用 Product 类。在 Product 类映射的表 tb_product 中建立外键 factoryid 关联 Factory 类的映射表 tb_factory 的主键 factoryid，两个表的关联关系如图 16.15 所示。

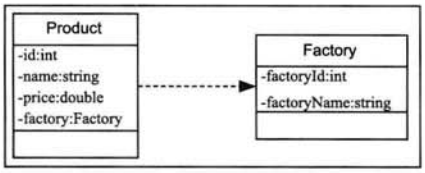


图 16.14 Product 类和 Factory 类的依赖关系

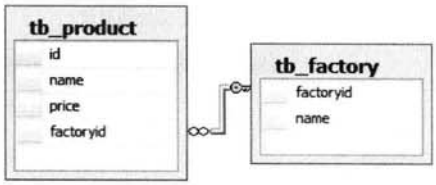


图 16.15 两个表的关联关系

【例 16.9】 建立产品对象与生产商对象的多对一单向关联，并利用映射关系查询完整的图书信息。

实例位置：光盘 \MR\ 源码 \ 第 16 章 \16.9

在产品对象的映射文件中建立多对一的关联，代码如下：

```
<!-- 产品信息字段配置信息 -->
<hibernate-mapping>
  <class name="com.wgh.product.Product" table="tb_product1">
  <!-- 指定的缓存的同步策略 -->
  <cache usage="read-only"/>
  <id name="id" column="id" type="int">                    <!-- id 值 -->
              <generator class="native"/>
        </id>
        <!-- 产品名称 -->
        <property name="name" type="string" length="45">
              <column name="name"/>
        </property>
        <!-- 产品价格 -->
        <property name="price" type="double">
              <column name="price"/>
        </property>
        <!-- 多对一关联映射 -->
        <many-to-one name="factory" class="com.wgh.factory.Factory">
        <!-- 映射的字段 -->
        <column name="factoryid"/>
        </many-to-one>
        </class>
</hibernate-mapping>
```

在上面的代码中，<many-to-one> 元素定义两个持久化类的关联，这种关联是数据表间的

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

多对一关联，需要此持久化类映射表的外键引用另一个持久化类映射表的主键，即映射的字段。其中 name 属性的值是持久化类中的属性，class 属性是关联的目标持久化类。

创建一个 Servlet，在其 doPost() 方法中编写查询完整图书信息的代码，关键代码如下：

```
Session session = null;                                          // 声明第一个 Session 对象
try {
    //Hibernate 的持久化操作
    session = HibernateUtil.getSession();                        // 获取 Session
    session.beginTransaction();                                  // 事务开启
    Product product = (Product) session.get(Product.class, new Integer("1"));
                                                                 // 装载对象
    System.out.println("产品名称: "+product.getName());
    System.out.println("产品价格: "+product.getPrice()+"元");
    System.out.println("生产商: "+product.getFactory().getFactoryName());
    session.getTransaction().commit();                           // 事务提交
} catch (Exception e) {
    e.printStackTrace();
    session.getTransaction().rollback();                         // 事务回滚
} finally{
    HibernateUtil.closeSession();                                // 关闭 Session
}
```

说明　由于 Product 类引用了 Factory 类，所以在产品实例中可以调用 Factory 类中的 getFactoryName() 方法获取生产商的名称。

程序运行后，控制台输出的信息如图 16.16 所示。

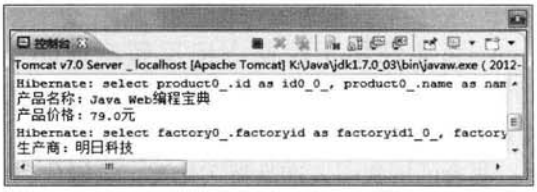


图 16.16　通过多对一单向关联获得的产品详细信息

从控制台输出的信息可以看到当查询生产商名称时，Hibernate 自动输出了一条语句查询。单向多对一关联时，只能通过主控方对被动方执行级联更新，即想要获取某件商品的生产商信息必须首先加载该产品的持久化对象。

16.5.3　理解并配置多对一双向关联

在执行双向多对一的关联时，Hibernate 既可以通过主控方实体加载被控方的实体，也可以通过被控方实体加载对应的主控方实体。即在单向一对多的基础上，在被控方（Product 类）中配置与主控方（Factory 类）对应的多对一关系。本小节仍将以生产商对象（Factory 类）与产品对象（Product 类）为例讲解 Hibernate 的多对一双向关联，二者的关联关系如图 16.17 所示。

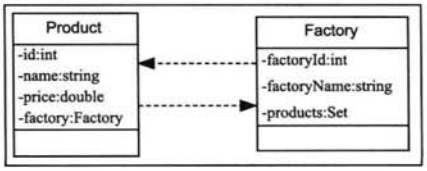


图 16.17　Factory 类与 Product 类的关联关系

由于一个生产商可能对应多件产品，所以在类 Factory 中以集合 Set 的方式引入产品对象，在

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

映射文件中通过 <set> 标签进行映射。

【例 16.10】 在原来产品对象与生产商对象的多对一单向关联基础上建立双向的多对一关系，并利用映射关系通过生产商查询对应的图书信息。

实例位置：光盘 \MR\ 源码 \ 第 16 章 \16.10

由于是在例 16.9 的基础上做出的新的映射关系，所以修改 Factory 类及其对应的映射文件即可，其映射文件的配置代码如下：

```
<!-- 产品信息字段配置信息 -->
<hibernate-mapping>
   <class name="com.wgh.factory.Factory" table="tb_factory">
          <!-- id值 -->
          <id name="factoryId" column="factoryid" type="int">
                 <generator class="native"/>
          </id>
          <!-- 生产商名称 -->
          <property name="factoryName" type="string" length="45">
                 <column name="factoryname"/>
          </property>
          <!-- 定义一对多映射 -->
          <set name="products" inverse="true">
                 <!--用于指定关联的id，这里指Product对象中关联的Factory对象的id，并不
                 是Product对象的id-->
                 <key column="factoryid" />
                 <one-to-many class="com.wgh.product.Product" />
          </set>
   </class>
</hibernate-mapping>
```

在上面的代码中，inverse 属性用于控制方向反转，将 inverse 属性设置为 true，表示 Factory 对象不再是主控方，而是将关联关系的维护交给关联对象 Product 类来完成，在 Product 对象持久化时会主动获取关联的 Factory 类的 id。

在生产商持久化类中以集合的形式引入产品持久化类，关键代码如下：

```
public class Factory {
   private Integer factoryId;                          // 生产商的 id
   private String factoryName;                         // 生产商名称
   private Set<Product> products;                      //Set 集合，一个厂商所对应的所有图书
   …                                                   // 省略的 Getter 和 Setter 方法
}
```

创建一个 Servlet，在其 doPost() 方法中编写通过加载生产商对象查询关联的产品信息的代码，关键代码如下：

```
Session session = null;                                    // 声明一个 Session 对象
try {
       //Hibernate 的持久化操作
       session = HibernateUtil.getSession(); // 获取 Session
       session.beginTransaction();                         // 事务开启
       Factory factoty = (Factory) session.get(Factory.class, new
       Integer("1"));                                      // 加载对象
       System.out.println(" 生产商: "+factoty.getFactoryName());
                                                           // 打印生产商名称
       Set<Product> products = factoty.getProducts();
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                                                        // 获取集合对象
            // 通过迭代输出产品信息
            for (Iterator<Product> it = products.iterator(); it.hasNext();) {
                Product product = (Product) it.next();
                System.out.println("产品名称: " + product.getName()+"||
                产品价格: "+product.getPrice());
            }
            session.getTransaction().commit();    // 事务提交
        } catch (Exception e) {
            e.printStackTrace();
            session.getTransaction().rollback(); // 事务回滚
        } finally{
            HibernateUtil.closeSession();         // 关闭 Session
        }
    }
```

因为配置了产品对象与生产商对象的多对一单关联映射，所以 Factory 与 Product 类都持有对方的引用。在程序中加载生产商对象即可获取其关联的产品对象的集合，Hibernate 将把产品对象映射到 Set 集合中。通过 Java 的迭代器在控制台输出一个生产商所属的产品信息，本实例的运行结果如图 16.18 所示。

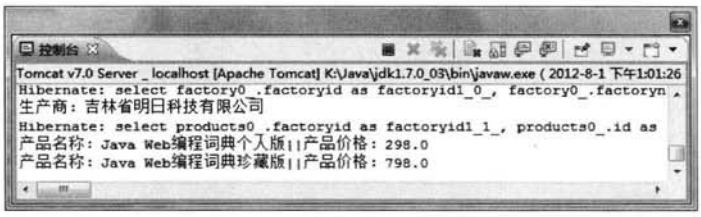


图 16.18　运行结果

16.5.4　理解并配置一对一主键关联

一对一的主键关联指两个表之间通过主键形成一对一的映射，如每个公民只允许拥有一个身份证，公民与身份证就是一对一的关系。定义 tb_people（公民表）和 tb_idcard（身份证表），其中 tb_people 表的 id 既是该表的主键，也是该表的外键，两个表之间的关联关系如图 16.19 所示。

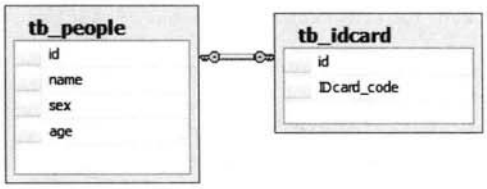


图 16.19　公民表与身份证表的关联关系

从关联关系可以看出只要程序知道一个表的信息即可获取另一个表的信息，即在 Hibernate 中两个表所映射的实体对象必然是互相引用的，建立的是双向一对一的主键关联关系。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

注意 在 Hibernate 中有单向和双向的一对一主键关联关系。

【例 16.11】 建立公民对象与身份证对象的一对一主键关联。

实例位置：光盘 \MR\ 源码 \ 第 16 章 \16.11

创建公民信息表的实体对象 People.java 和身份证的实体对象 IDcard.java，其中公民对象为主控方，身份证对象为被控方。两个实体对象之间的依赖关系如图 16.20 所示。

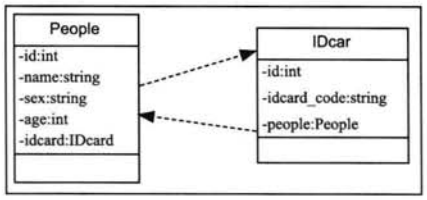


图 16.20 公民实体对象与身份证实体对象之间的依赖关系

在公民实体对象的映射文件中的关键代码如下：

```
<!-- 公民信息字段配置信息 -->
<hibernate-mapping>
  <class name="com.wgh.people.People" table="tb_people">
        <!-- id值 -->
        <id name="id" column="id" type="int">
              <generator class="native"/>
        </id>
        <!-- 公民姓名 -->
        <property name="name" type="string" length="45">
              <column name="name"/>
        </property>
        <!-- 公民性别 -->
        <property name="sex" type="string" length="2">
              <column name="sex"/>
        </property>
        <!-- 公民年龄 -->
        <property name="age" type="int">
              <column name="age"/>
        </property>
        <!-- 一对一映射 -->
        <one-to-one name="idcard" class="com.wgh.idcard.IDcard" cascade="all"/>
        </class>
</hibernate-mapping>
```

在身份证的实体对象映射文件中的主键需要参考公民实体对象的外键，关键代码如下：

```
<!-- 公民身份证字段信息配置信息 -->
<hibernate-mapping>
  <class name="com.wgh.idcard.IDcard" table="tb_idcard">
        <!-- id值 -->
        <id name="id" column="id" type="int">
              <!-- 外键生成 -->
              <generator class="foreign">
                    <param name="property">people</param>
              </generator>
        </id>
        <!-- 公民身份证号 -->
        <property name="idcard_code" type="string" length="45" not-null="true">
              <column name="IDcard_code"/>
        </property>
  </class>
</hibernate-mapping>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

16.5.5　级联操作

在数据库操作中常常利用主外键约束来保护数据库数据操作的一致性，如在公民表和身份证表的一对一关系中不允许单独删除公民表中的某条公民信息，必须同时删除两个表中关联的信息。即两个表的操作需要同步进行，在这种情况下需要 Hibernate 的级联（cascade）操作。

级联操作指的是当主控方执行 save、update 或 delete 操作时，关联对象（被控方）是否执行同步操作。在映射文件中通过设置 cascade 属性的参数决定是否对关联对象采用级联操作，其参数如表 16.4 所示。

表 16.4　cascade 属性的参数

参　数	说　明
all	所有情况下均采用级联操作
none	默认，所有情况下均不采用级联操作
save-update	在执行 save-update 方法时执行级联操作
delete	在执行 delete 方法时执行级联操作

说明　级联关系在 Hibernate 中占有重要的地位，可以保证主控方关联的被控方操作的一致性。

【例 16.12】 利用级联操作删除公民表及其在身份证表中关联的信息。

实例位置：光盘 \MR\ 源码 \ 第 16 章 \16.12

在公民实体对象的映射文件 People.hbm.xml 中一对一关联关系的设置中设置关联对象的级联操作，关键代码如下：

```
<!-- 公民信息字段配置信息 -->
<hibernate-mapping>
   <class name="com.wgh.people.People" table="tb_people">
          …<!-- 省略的配置信息 -->
          <!-- 一对一映射 -->
          <one-to-one name="idcard" class="com.wgh.idcard.IDcard" cascade
          ="delete"/>
   </class>
</hibernate-mapping>
```

创建类 Manager，在 main() 方法中利用 Session 的 delete() 方法删除加载的公民对象，关键代码如下：

```
Session session = null;                                          // 声明一个 Session 对象
   try {
          //Hibernate 的持久化操作
          session = HibernateUtil.getSession();                  // 获取 Session
          session.beginTransaction();                            // 事务开启
          People people = (People) session.get(People.class, new Integer("1"));
                                                                 // 加载对象
          session.delete(people);                                // 删除加载的公民对象
          session.getTransaction().commit();                     // 事务提交
          System.out.println("数据删除成功！");
   } catch (Exception e) {
          e.printStackTrace();
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            session.getTransaction().rollback();          //事务回滚
        } finally{
            HibernateUtil.closeSession();                 //关闭 Session
        }
```

程序运行后在控制台输出的信息如图 16.21 所示。

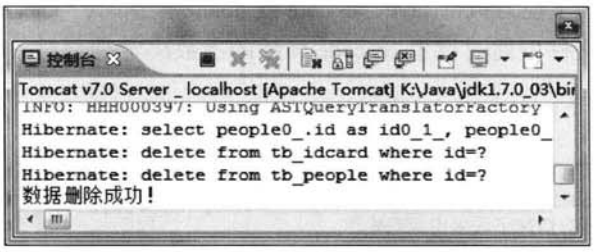


图 16.21　控制台输出的信息

在 main() 方法中只执行了一次 Session 的 delete() 方法，但是从控制台输出语句可以看出 Hibernate 执行了两次删除的操作，即同时删除了公民表和身份证表中关联的信息。第 1 条 SQL 语句是 Hibernate 加载公民实体对象时的 Select 语句，第 2 条与第 3 条语句是删除加载对象及其关联对象的 Delete 语句。

16.6　Hibernate查询语言

HQL（Hibernate Query Language）查询语言是完全面向对象的查询语言，它提供了面向对象的封装，可以理解如多态、继承和关联的概念。HQL 与 SQL 语句相似，但提供了更加强大的查询功能，它是 Hibernate 官方推荐的查询模式。

16.6.1　HQL 语言

HQL 与 SQL 的语句相似，其基本的使用方法也与 SQL 相同。由于 HQL 是面向对象的查询语言，所以需要从目标对象中查询信息并返回匹配单个实体对象或多个实体对象的集合，而 SQL 语句是从数据表中查找指定信息并返回单条信息或多个信息的集合。

注意　因为 HQL 是面向对象的查询语句，其查询目标是实体对象，即 Java 类。Java 类区分大小写，如 com.mr.Test 与 com.mr.TeSt 表示两个不同的类，所以 HQL 也区分大小写。

HQL 的基本语法格式如下：

```
select "对象.属性名"
from "对象"
where "过滤条件"
group by "对象.属性名"
having "分组条件"
order by "对象.属性名"
```

如实际应用中的 HQL 语句如下：

```
select * from Employee emp where emp.flag='1'
```

该语句等价于：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
from Employee emp where emp.flag='1'
```

该 HQL 语句查询过滤从数据库信息返回的实体对象的集合，其过滤条件为对象属性“Flag”为 1 的实体对象，其中 Employee 为实体对象。

16.6.2　查询实体对象

在 HQL 语句中，可以通过 from 子句直接查询实体对象，如：

```
from Person
```

在大多数情况下最好为查询的实体对象指定一个别名，方便在查询语句的其他地方引用实体对象，别名的命名方法如下：

```
from Person per
```

说明　别名的首字母最好为小写，这是HQL语句的规范写法。与Java中变量的命名规则一致，可以避免与语句中的实体对象混淆。

上面的 HQL 语句将查询数据库中实体对象 Person 对应的所有数据，并以封装好的 Person 对象的集合形式返回。但是该语句的局限性是查询实体对象 Person 映射的所有数据库字段，相当于 SQL 语句中的“Select *”。在 HQL 中需要通过动态实例化查询来实现这个功能，如：

```
select new Person(id,name) from Person per
```

此种查询方式通过 new 关键字动态实例化实体对象，重新封装指定的实体对象属性。既不失去数据的封装性，又可提高查询的效率。

注意　在上面的语句中最好不要使用以下语句查询：

```
select per.id,per.name from Person per
```

因为此语句返回的不是原有的对象实体状态，而是一个 object 类型的数组，它破坏了数据的原有封装性。

【例 16.13】　查询 Employee 对象中的所有信息，并将查询结果显示在页面中。

实例位置：光盘 \MR\ 源码 \ 第 16 章 \16.13

创建 Servlet 并在其中通过 HQL 语句查询 Employee 对象的所有信息，关键代码如下：

```
List<Employee> emplist = new ArrayList<>();          // 实例化 List 信息集合
Session session = null;                               // 实例化 session 对象
try {
    session = HibernateUtil.getSession();             // 获得 session 对象
    String hql = "from Employee emp";                 // 查询 HQL 语句
    Query q = session.createQuery(hql);               // 执行查询操作
    emplist = q.list();                               // 将返回的对象转化为 List 集合
} catch (HibernateException e) {
    e.printStackTrace();
} finally {
    HibernateUtil.closeSession();                     // 关闭 session
}
```

当查询结果返回页面后，利用 JSTL 显示查询的列表信息，程序运行结果如图 16.22 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

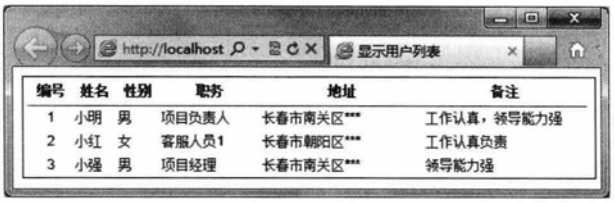


图 16.22 查询全部员工信息

16.6.3 条件查询

条件查询在实际应用中比较广泛，通常用其过滤数据库返回的查询数据，因为一个表中的所有数据并非一定都对用户有意义。

HQL 的条件查询与 SQL 语句一样都通过 where 子句实现，如在例 16.13 中查询性别为“男”的员工，HQL 语句如下：

```
from Employee emp where emp.sex="男"
```

修改例 16.13 中的 HQL 语句，页面的输出结果如图 16.23 所示。

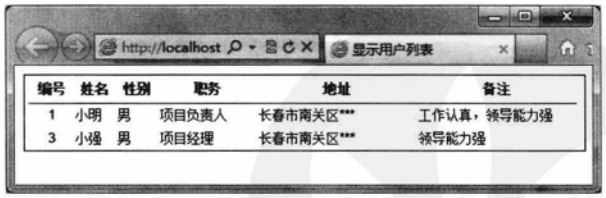


图 16.23 查询性别为“男”的员工

16.6.4 HQL 参数绑定机制

参数绑定机制可以使查询语句和参数值相互独立，不但可以提高程序的开发效率，还可以有效地防止 SQL 的注入攻击。在 JDBC 中的 PreparedStatement 对象通过动态赋值的形式绑定 SQL 语句的参数。在 HQL 中同样提供了动态赋值的功能，有如下两种不同的实现方法。

（1）利用顺序占位符“？”替代参数

在 HQL 语句中可以通过顺序占位符“？”替代参数值，利用 Query 对象的 setParameter() 方法为其赋值，此种操作方式与 JDBC 中的 PreparedStatement 对象的参数绑定方式相似。如在例 16.13 中查询性别为“男”的员工信息，代码如下：

```
session = HibernateUtil.getSession();                    // 获得 session 对象
String hql = "from Employee emp where emp.sex=?";        // 查询 HQL 语句
Query q = session.createQuery(hql);                      // 执行查询操作
q.setParameter(0, "男");                                 // 为占位符顺序赋值
emplist = q.list();
```

（2）利用引用占位符“:parameter”替代具体参数

“:parameter”中的引用占位符是“：”号与自定义参数名的组合。如在例 16.13 中查询性别为“男”的员工信息，代码如下：

```
session = HibernateUtil.getSession();                    // 获得 session 对象
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
String hql = "from Employee emp where emp.sex=:sex";     // 查询 HQL 语句
Query q = session.createQuery(hql);                       // 执行查询操作
q.setParameter("sex", "男");                              // 为引用占位符赋值
emplist = q.list();
```

16.6.5　排序查询

在 SQL 中通过 order by 子句、asc 和 desc 关键字实现查询结果集的排序操作，其中，asc 是正序排列，desc 是降序排列。在 HQL 查询语言中也提供了此功能，用法与 SQL 语句类似，只是排序的条件参数为实体对象的属性。

例如，按照年龄的倒序排列员工信息：

```
from Employee emp order by emp.age desc
```

按照 ID 的正序排列员工信息：

```
from Employee emp order by emp.id asc
```

16.6.6　应用聚合函数

在 HQL 查询语言中支持 SQL 中常用的聚合函数，如 sum、avg、count、max 和 min 等，其用法与 SQL 中基本相同。

例如，计算所有员工的平均年龄可以编写下面的 HQL 语句：

```
select avg(emp.age) from Employee emp
```

再例如，查询所有员工中年龄最小的员工信息可以编写下面的 HQL 语句：

```
select min(emp.age) from Employee emp
```

16.6.7　分组方法

在 HQL 中可以在 group by 子句中使用 having 语句，但是需要底层数据库的支持，如 MySQL 数据库不支持 having 语句。

【例 16.14】 分组统计男女员工的人数。

实例位置：光盘 \MR\ 源码 \ 第 16 章 \16.14

创建 Servlet 并在其 doPost() 方法中，利用 HQL 语句统计男女员工的人数，关键代码如下：

```
Session session = null;                                   // 实例化 session 对象
try {
   session = HibernateUtil.getSession();                  // 获得 session 对象
  String hql = "select emp.sex,count(*) from Employee emp group by emp.sex";
                                                          // 条件查询 HQL 语句
    Query q = session.createQuery(hql);                   // 执行查询操作
    List<Employee> emplist = q.list();
    Iterator it = emplist.iterator();                     //使用迭代器输出返回的对象数组
  while(it.hasNext()) {
         Object[] results = (Object[])it.next();
           System.out.print("员工性别：" + results[0] + "————");
           System.out.println("人数：" + results[1]);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        }
    } catch (HibernateException e) {
        e.printStackTrace();
    } finally {
        HibernateUtil.closeSession();                    // 关闭 session
    }
```

注意 在 group by 与 order by 子句中都不能含有算术表达式，并且分组条件不能是实体对象本身，如 group by Employee 是错误的，除非实体对象的所有属性都是非聚集的。

程序运行后控制台输出的信息如图 16.24 所示。

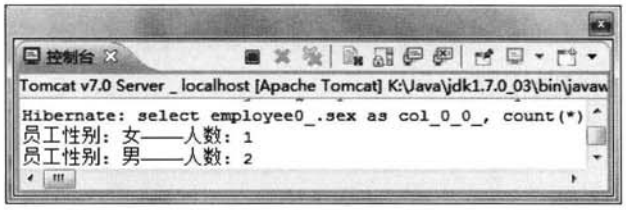


图 16.24　分组统计员工信息

16.6.8　联合查询

联合查询是执行数据库多表操作时必不可少的操作之一，如在 SQL 中连接查询方式中的内连接查询（inner join）、左连接查询（left outer join）、右连接查询（right outer join）和全连接查询（full join）。在 HQL 查询语句中也支持联合查询方式。

例如，在公民表与身份证表中一对一映射关系即可通过 HQL 左连接的方式获取关联的信息。

【例 16.15】 通过 HQL 的左连接查询获取公民信息及其关联的身份证信息。

实例位置：光盘 \MR\ 源码 \ 第 16 章 \16.15

创建 Servlet 并在其中利用左连接查询获取公民信息和其关联的身份证信息，关键代码如下：

```
Session session = null;                                  // 实例化 session 对象
List<Object[]> list = new ArrayList<Object[]>();
try {
    session = HibernateUtil.getSession();                // 获得 session 对象
    session.beginTransaction();                          // 开启事务
    // 条件查询 HQL 语句
    String hql = "select peo.id,peo.name,peo.age,peo.sex,c.idcard_code from
    People peo left join peo.idcard c";
    Query q = session.createQuery(hql);                  // 执行查询操作
    list = q.list();
    session.getTransaction().commit();                   // 提交事务
} catch (HibernateException e) {
    e.printStackTrace();
    session.getTransaction().rollback();                 // 出错将回滚事务
} finally {
    HibernateUtil.closeSession();                        // 关闭 session
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

程序运行后页面输出的信息如图 16.25 所示。



图 16.25　利用左连接查询公民信息

16.7　上机实践

16.7.1　修改员工信息

➤➤➤ 题目描述

本实例主要综合应用 Hibernate 中的功能，要求可以查看或修改数据并返回修改后的内容，其中的 3 个页面如图 16.26 ～图 16.28 所示。

➤➤➤ 技术指导

HQL 语句是 Hibernate 的面向对象查询语句，可以用于获取所有员工信息，每个员工信息以实例对象的形式返回。HQL 的语法类似 SQL，它需要使用 Session 的 createQuery() 方法查询。与 SQL 语句的不同是，将 SQL 中的表名更换为映射的持久化对象名。

另外，还可以使用 list() 方法将获取的所有的员工信息对象转换为 list。

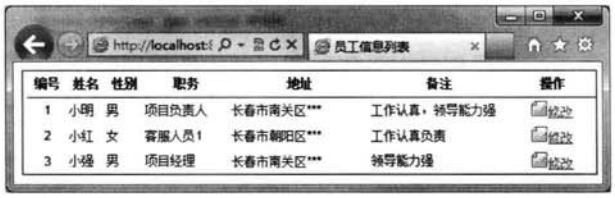


图 16.26　员工列表页面

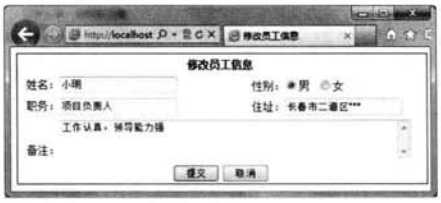


图 16. 27　修改员工信息页面

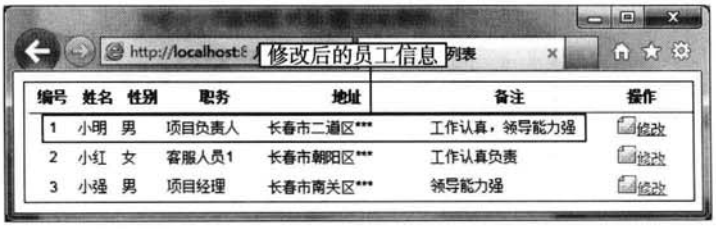


图 16.28　修改后返回的列表页面

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

16.7.2 员工信息的模糊查询

➤➤➤ 题目描述

模糊查询是一种常用的数据库查询手段，可以通过通配符模糊匹配查询的数据。如查询姓王的员工信息，只需在姓名查询条件中输入“王”字。本实例应用模糊查询，在浏览器的地址栏中输入程序的 URL 地址，默认打开的页面如图 16.29 所示。

在“搜索员工信息”文本框中输入“红”字，单击“搜索员工信息”按钮，如图 16.30 所示。

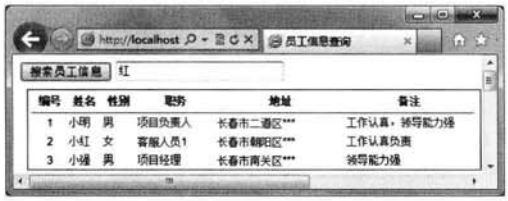


图 16.29 默认打开的页面

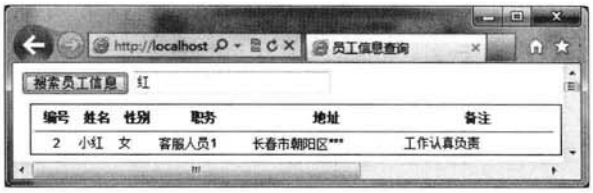


图 16.30 通过模糊查询的员工信息

➤➤➤ 技术指导

模糊查询的实现比较简单，关键是编写符合需求的 HQL 语句。

HQL 与 SQL 模糊查询语句的使用方式基本相同，同样使用 like 子句。

由于在使用模糊查询时需要使用通配符“%”进行匹配，所以在赋值时要加上“%”符号。

16.7.3 按年龄升序显示公民信息

➤➤➤ 题目描述

本章介绍了通过 Hibernate 操作数据库的相关内容。本上机实践尝试编写一个程序，要求通过 Hibernate 操作数据库，实现按年龄升序显示公民信息，当年龄相同时，按性别降序排列，运行效果如图 16.31 所示。



图 16.31 按年龄升序显示公民信息

➤➤➤ 技术指导

在 HQL 查询语言中，排序的方法与 SQL 相似，它也使用“order by”子句对查询结果集进行排序，并且使用 asc 或 desc 关键字指定排序方式，但要注意 HQL 是面向对象查询语言，要严格区分对象与属性的大小写。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

本上机实践的关键代码参考如下：

```
String hql = "from People p order by p.age asc,p.sex desc";
list=session.createQuery(hql).list();
```

16.8　小结

本章详细讲解了 Hibernate 的基础知识及其高级应用，其持久化操作是开发应用系统基础，熟练掌握 Hibernate 的基础知识能够为快速开发应用程序打下坚实的基础。高级应用中实体关联关系的映射在 Hibernate 中是最难配置正确的，掌握好该配置是本章的重点。本章还介绍了 Hibernate 的另外一种数据查询方式——HQL 语言，可以将其理解为面向对象的 SQL 语句。不同的是其查询目标是实体对象，不是数据表，而使用方式与传统的 SQL 语句基本相同。

16.9　习题

选择题

1. Hibernate 配置文件中，配置数据库驱动使用的是（　　）属性。
 A. connection.driver_class　　B. connection.url　　C. dialect
2. Hibernate 配置文件中，配置连接数据库驱动的用户名使用的是（　　）属性。
 A. connection.driver_class　　B. connection.username　　C. dialect
3. Hibernate 配置文件中，配置 Hibernate 方言使用的是（　　）属性。
 A. connection.driver_class　　B. connection.url　　C. dialect
4. 要实现多对一关联映射，可以使用元素（　　）。
 A. <many-to-one>　　B. <property>　　C. <class>
5. 要设置延迟检索策略，可以将 Lazy 属性设置为（　　）。
 A. false　　B. true　　C. yes

填空题

1. Hibernate 配置文件中的 <mapping resource/> 是用来对_____进行配置。
2. 映射 Java 基本类型为 String 的属性，Hibernate 的映射类型为_____。
3. 要实现查询数据，可使用 session 的_____方法。
4. Hibernate 映射文件中的 <class> 元素是用来对_____进行映射的。
5. 在 HQL 语法中，要实现排序查询，可以使用子句_____。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 17 章　简化企业开发瑰宝——Spring 技术

视频讲解：72 分钟

在 Java EE 开发平台中，Spring 是一种优秀的轻量级企业应用解决方案，它倡导一切从实际出发，核心技术是 IoC（Inversion of Control，控制反转）和 AOP（Aspect Oriented Programming，面向切面编程）技术。它为 Java 带来了一种全新的编程思想，其目的是解决企业应用开发的复杂性。本章将对 Spring 技术进行介绍。

通过学习本章，读者可以达到以下学习目的：

- 了解 Spring 技术；
- 掌握 Spring 的获取及其配置；
- 掌握依赖注入；
- 掌握 Bean 的作用域及其特殊处理；
- 了解 AOP；
- 掌握 Spring 的切入点；
- 掌握 Aspect 对 AOP 的支持；
- 掌握 Spring 持久化。

17.1　Spring概述

Spring 是一个开源框架，由 Rod Johnson 创建，从 2003 年年初正式启动。它能够降低开发企业应用程序的复杂性，使用 Spring 替代 EJB（Enterprise Java Beans）开发企业级应用，而不用担心工作量太大、开发进度难以控制和复杂的测试过程等问题。Spring 简化了企业应用的开发、降低了开发成本并整合了各种流行框架，它以 IoC 和 AOP 两种先进的技术为基础完美地降低了企业级开发的复杂度。

17.1.1　Spring 的组成

Spring 框架主要由 7 大模块组成，它们提供了企业级开发需要的所有功能。每个模块都可以单独使用，也可以和其他模块组合使用，灵活且方便地部署模块可以使开发的程序更加简洁灵活。图 17.1 所示是 Spring 的 7 大模块。

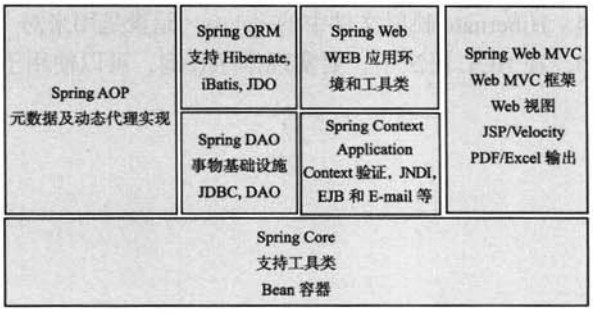


图 17.1　Spring 的 7 大模块

（1）Core 模块

该模块是 Spring 的核心容器，它实现了 IoC 模式和 Spring 框架的基础功能。在模块中包含的最重要的 BeanFactory 类是 Spring 的核心类，负责配置与管理 JavaBean。它采用 Factory 模式实现了 IoC 容器，即依赖注入。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（2）Context 模块

该模块继承 BeanFactory（或者说 Spring 核心）类，并且添加了事件处理、国际化、资源加载、透明加载以及数据校验等功能。它还提供了框架式的 Bean 的访问方式和很多企业级的功能，如 JNDI 访问、支持 EJB、远程调用、集成模板框架、E-mail 和定时任务调度等。

（3）AOP 模块

Spring 集成了所有 AOP 功能，通过事务管理可以将任意 Spring 管理的对象 AOP 化。Spring 提供了用标准 Java 语言编写的 AOP 框架，其中大部分内容都是根据 AOP 联盟的 API 开发。它使应用程序抛开了 EJB 的复杂性，但拥有传统 EJB 的关键功能。

（4）DAO 模块

该模块提供了 JDBC 的抽象层，简化了数据库厂商的异常错误处理（不再从 SQLException 继承大批代码），大幅度减少了代码的编写并且提供了对声明式和编程式事务的支持。

（5）ORM 映射模块

该模块提供了对现有 ORM 框架的支持，各种流行的 ORM 框架已经非常成熟，并且拥有大规模的市场（如 Hibernate）。Spring 没有必要开发新的 ORM 工具，但是为 Hibernate 提供了完美的整合功能，并且支持其他 ORM 工具。

（6）Web 模块

该模块建立在 Spring Context 基础之上，提供了 Servlet 监听器的 Context 和 Web 应用的上下文，为现有的 Web 框架（如 JSF、Tapestry 和 Struts 等）提供了集成。

（7）MVC 模块

该模块建立在 Spring 核心功能之上，使其拥有 Spring 框架的所有特性。从而能够适应多种视图、模板技术、国际化和验证服务，实现控制逻辑和业务逻辑的清晰分离。

17.1.2　获取 Spring

在使用 Spring 之前必须首先在 Spring 的官方网站免费下载 Spring 工具包，其网址为 http://www.springsource.org/download。在该网站可以免费获取 Spring 的帮助文档和 jar 包，本书中的所有实例使用的 Spring 的 jar 包的版本为 spring-framework-3.1.1.RELEASE。

将 spring.jar 包和 dist 目录下的所有的 jar 包导入到项目中，随后即可开发 Spring 的项目。

注意　不同版本之间的 jar 包可能会存在不同，所以读者应尽量保证使用与本书版本一致的 jar 包。

17.1.3　配置 Spring

获得并打开 Spring 的发布包之后，其 dist 目录中包含 Spring 的 20 个 jar 文件，其相关功能说明如表 17.1 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 17.1　Spring 的 jar 包相关功能说明

jar 包的名称	说　明
org.springframework.aop-3.1.1.RELEASE.jar	Spring 的 AOP 模块
org.springframework.asm-3.1.1.RELEASE.jar	Spring 独立的 asm 程序，相比 2.5 版本，需要额外的 asm.jar 包
org.springframework.aspects-3.1.1.RELEASE.jar	Spring 提供的对 AspectJ 框架的整合
org.springframework.beans-3.1.1.RELEASE.jar	Spring 的 IoC 的基础实现
org.springframework.context.support-3.1.1.RELEASE.jar	Spring 上下文的扩展支持，用于 MVC 方面
org.springframework.context-3.1.1.RELEASE.jar	Spring 的上下文，spring 提供在基础 IoC 功能上的扩展服务，此外还提供许多企业级服务的支持，如邮件服务、任务调度、JNDI 定位、EJB 集成、远程访问、缓存以及各种视图层框架的封装等
org.springframework.core-3.1.1.RELEASE.jar	Spring 的核心模块
org.springframework.expression-3.1.1.RELEASE.jar	Spring 的表达式语言
org.springframework.instrument.tomcat-3.1.1.RELEASE.jar	Spring 对 Tomcat 连接池的支持
org.springframework.instrument-3.1.1.RELEASE.jar	Spring 对服务器的代理接口
org.springframework.jdbc-3.1.1.RELEASE.jar	Spring 的 JDBC 模块
org.springframework.jms-3.1.1.RELEASE.jar	Spring 为简化 JMS API 使用而做的简单封装
org.springframework.orm-3.1.1.RELEASE.jar	Spring 的 ORM 模块，支持 Hibernate 和 JDO 等 ORM 工具
org.springframework.oxm-3.1.1.RELEASE.jar	Spring 对 Object/XMI 的映射的支持，可以在 Java 与 XML 之间来回切换
org.springframework.test-3.1.1.RELEASE.jar	Spring 对 Junit 等测试框架的简单封装
org.springframework.transaction-3.1.1.RELEASE.jar	Spring 为 JDBC、Hibernate、JDO、JPA 等提供的一致的声明式和编程式事务管理
org.springframework.web.portlet-3.1.1.RELEASE.jar	Spring MVC 的增强
org.springframework.web.servlet-3.1.1.RELEASE.jar	Spring 对 Java EE6.0 和 Servlet 3.0 的支持
org.springframework.web.struts-3.1.1.RELEASE.jar	整合 Struts
org.springframework.web-3.1.1.RELEASE.jar	Sping 的 Web 模块，包含 Web application context

除了表 17-1 中给出的这些 jar 包以外，Spring 还需要 commons-logging.jar 和 aopalliance.jar 包的支持。其中，commons-logging.jar 包可以到 http://commons.apache.org/logging/ 网站下载；aopalliance.jar 包可以到 http://sourceforge.net/projects/aopalliance/files/ 网站下载。

得到这些包以后，我们可以将其放在应用 Spring 的 Web 项目的 WEB-INF 文件夹下的 lib 文件夹中，Web 服务器启动时会自动加载 lib 中的所有 jar 文件。在使用 Eclipse 开发工具时，

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

我们也可以将这些包配置为一个用户库，然后在需要应用 Spring 的项目中加载这个用户库就可以了。

Spring 的配置结构如图 17.2 所示。

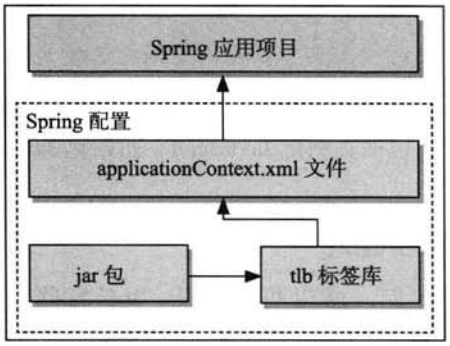


图 17.2　Spring 的配置结构

17.1.4　使用 BeanFactory 管理 Bean

BeanFactory 采用了 Java 经典的工厂模式，通过从 XML 配置文件或属性文件（.properties）中读取 JavaBean 的定义来创建、配置和管理 JavaBean。BeanFactory 有很多实现类，其中 XmlBeanFactory 可以通过流行的 XML 文件格式读取配置信息来加载 JavaBean。BeanFactory 在 Spring 中的作用如图 17.3 所示。

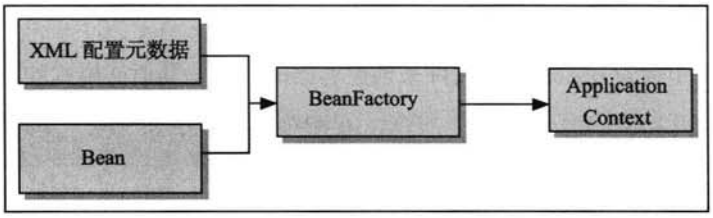


图 17.3　BeanFactory 在 Spring 中的作用

例如，加载 Bean 配置的代码如下：

```
Resource resource = new ClassPathResource("applicationContext.xml");
                                                          // 加载配置文件
BeanFactory factory = new XmlBeanFactory(resource);
Test  test = (Test) factory.getBean("test");              // 获取 Bean
```

ClassPathResource 读取 XML 文件并将参数传递给 XmlBeanFactory，applicationContext.xml 文件的代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.spring framework.
org/dtd/spring- beans.dtd">
<beans>
   <bean id="test" class="com.mr.test.Test"/>
</beans>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在 <beans> 标签中定义 JavaBean 的名称和类型，在程序代码中利用 BeanFactory 的 getBean() 方法获取 JavaBean 的实例并且向上转换为需要的接口类型，从而在容器中开始这个 JavaBean 的生命周期。

说明 BeanFactory 在调用 getBean() 方法之前不会实例化任何对象，只有在需要创建 JavaBean 的实例对象时才会为其分配资源空间。这使其更适合物理资源受限制的应用程序，尤其是内存受限制的环境。

Spring 中 Bean 的生命周期包括实例化 JavaBean、初始化 JavaBean、使用 JavaBean 和销毁 JavaBean 共 4 个阶段。

17.1.5 应用 AplicationContext

BeanFactory 实现了 IoC 控制，所以可以称为“IoC 容器”，而 ApplicationContext 扩展了 BeanFactory 容器并添加了对 I18N（国际化）和生命周期事件的发布监听等更加强大的功能，使之成为 Spring 中强大的企业级 IoC 容器。在这个容器中提供了对其他框架和 EJB 的集成、远程调用、WebService、任务调度和 JNDI 等企业服务，在 Spring 应用中大多采用 ApplicationContext 容器来开发企业级的程序。

说明 ApplicationContext 不仅提供了 BeanFactory 的所有特性，而且也允许使用更多的声明方式来得到所需的功能。

ApplicationContext 接口有如下 3 个实现类，可以实例化其中任何一个类来创建 Spring 的 ApplicationContext 容器。

（1）ClassPathXmlApplicationContext 类

从当前类路径中检索并加载配置文件来创建容器的实例，其语法格式如下：

```
ApplicationContext context=new ClassPathXmlApplicationContext(String config
Location);
```

configLocation 参数指定 Spring 配置文件的名称和位置。

（2）FileSystemXmlApplicationContext 类

该类不从类路径中获取配置文件，而是通过参数指定配置文件的位置。它可以获取类路径之外的资源，其语法格式如下：

```
ApplicationContext context=new FileSystemXmlApplicationContext(String config
Location);
```

（3）WebApplicationContext 类

WebApplicationContext 是 Spring 的 Web 应用容器，在 Servlet 中使用该类的方法有两种：一是在 Servlet 的 web.xml 文件中配置 Spring 的 ContextLoaderListener 监听器；二是修改 web.xml 配置文件，在其中添加一个 Servlet，定义使用 Spring 的 org.springframework.web.context.Context LoaderServlet 类。

说明 JavaBean 在 ApplicationContext 和 BeanFactory 容器中的生命周期基本相同，如果在 JavaBean 中实现了 ApplicationContextAware 接口，容器会调用 JavaBean 的 setApplicationContext() 方法将容器本身注入到 JavaBean 中，使 JavaBean 包含容器的应用。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

17.2　依赖注入

Spring 框架中的各个部分充分使用了依赖注入（Dependency Injection）技术，使得代码中不再有单实例垃圾和麻烦的属性文件，取而代之的是一致和优雅的程序应用代码。

17.2.1　控制反转与依赖注入

使程序组件或类之间尽量形成一种松耦合的结构，开发人员在使用类的实例之前需要创建对象的实例，IoC 将创建实例的任务交给 IoC 容器，这样开发应用代码时只需要直接使用类的实例，这就是 IoC 控制反转。通常用一个所谓的好莱坞原则（Don't call me. I will call you，请不要给我打电话，我会打给你）来比喻这种控制反转的关系。Martin Fowler 曾专门写了一篇文章“Inversion of Control Containers and the Dependency Injection pattern”讨论控制反转这个概念，并提出一个更为准确的概念，即“依赖注入”。

依赖注入有如下 3 种实现类型，Spring 支持后两种。

（1）接口注入

该类型基于接口将调用与实现分离，这种依赖注入方式必须实现容器所规定的接口，使程序代码和容器的 API 绑定在一起，这不是理想的依赖注入方式。

（2）Setter 注入

该类型基于 JavaBean 的 Setter 方法为属性赋值，在实际开发中得到了最广泛的应用（其中很大一部分得益于 Spring 框架的影响），如：

```
public class User {
    private String name;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

在上述代码中定义了一个字段属性 name 并且使用 Getter 和 Setter 方法，这两个方法可以为字段属性赋值。

（3）构造器注入

该类型基于构造方法为属性赋值，容器通过调用类的构造方法将其所需的依赖关系注入其中，如：

```
public class User {
    private String name;
    public User(String name){                    //构造器
        this.name=name;                          //为属性赋值
    }
}
```

在上述代码中使用构造方法为属性赋值，这样做的好处是在实例化类对象的同时完成了属性的初始化。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明 由于在控制反转模式下把对象放在 XML 文件中定义，所以开发人员实现一个子类更为简单，即只需要修改 XML 文件。而且控制反转颠覆了“使用对象之前必须创建”的传统观念，开发人员不必再关注类是如何创建的，只需从容器中抓取一个类后直接调用即可。

17.2.2 配置 Bean

在 Spring 中无论使用哪种容器，都需要从配置文件中读取 JavaBean 的定义信息，然后根据定义信息创建 JavaBean 的实例对象并注入其依赖的属性。由此可见 Spring 中所谓的配置主要是对 JavaBean 的定义和依赖关系而言，JavaBean 的配置也针对配置文件。

要在 Spring IoC 容器中获取一个 bean，首先要在配置文件中的 <beans> 元素中配置一个子元素 <bean>，Spring 的控制反转机制会根据 <bean> 元素的配置来实例化这个 bean 实例。

如配置一个简单的 JavaBean：

```
<bean id="test" class="com.mr.Test"/>
```

其中，id 属性为 bean 的名称；class 属性为对应的类名，这样通过 BeanFactory 容器的 getBean("test") 方法即可获取该类的实例。

17.2.3 Setter 注入

一个简单的 JavaBean 的最明显规则是一个私有属性对应 Setter 和 Getter 方法以封装属性。既然 JavaBean 有 Setter 方法来设置 Bean 的属性，Spring 就会有相应的支持。配置文件中的 <property> 元素可以为 JavaBean 的 Setter 方法传递参数，即通过 Setter 方法为属性赋值。

【例 17.1】 通过 Spring 的赋值为用户 JavaBean 的属性赋值。

实例位置：光盘 \MR\ 源码 \ 第 17 章 \17.1

首先创建用户的 JavaBean，关键代码如下：

```
public class User {
    private String name;                    // 用户姓名
    private Integer age;                    // 年龄
    private String sex;                     // 性别
    …                                       // 省略的 Setter 和 Getter 方法
}
```

在 Spring 的配置文件 applicationContext.xml 中配置该 JavaBean，关键代码如下：

```
<!-- User Bean -->
<bean name="user" class="com.mr.user.User">
   <property name="name">
          <value> 无语 </value>
   </property>
   <property name="age">
          <value>30</value>
   </property>
   <property name="sex">
          <value> 女 </value>
   </property>
</bean>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在上面的代码中，<value> 标签用于为 name 属性赋值，这是一个普通的赋值标签。直接在成对的 <value> 标签中放入数值或其他赋值标签，Spring 会把这个标签提供的属性值注入到指定的 JavaBean 中。

说明　如果 JavaBean 的某个属性是 List 集合或数组类型，则需要使用 <list> 标签为 List 集合或数组类型的每一个元素赋值。

创建类 Manager，在其 main() 方法中的关键代码如下：

```
Resource resource = new ClassPathResource("applicationContext.xml");
                                                          // 加载配置文件
BeanFactory factory = new XmlBeanFactory(resource);
User user = (User) factory.getBean("user");               // 获取 Bean
System.out.println("用户姓名—"+user.getName());            // 输出用户的姓名
System.out.println("用户年龄—"+user.getAge());             // 输出用户的年龄
System.out.println("用户性别—"+user.getSex());             // 输出用户的性别
```

程序运行后控制台输出的信息如图 17.4 所示。

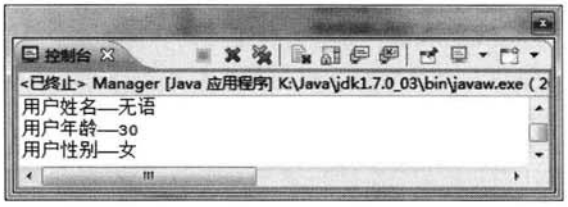


图 17.4　控制台输出的信息

17.2.4　构造器注入

在类被实例化时其构造方法被调用并且只能调用一次，所以构造器常被用于类的初始化操作。<constructor-arg> 是 <bean> 元素的子元素，通过 <constructor-arg> 元素的 <value> 子元素可以为构造方法传递参数。

【例 17.2】　通过 Spring 的构造器注入为用户 JavaBean 的属性赋值。

实例位置：光盘 \MR\ 源码 \ 第 17 章 \17.2

在用户 JavaBean 中创建构造方法，代码如下：

```
public class User {
    private String name;                                  // 用户姓名
    private Integer age;                                  // 年龄
    private String sex;                                   // 性别
    // 构造方法
    public User(String name,Integer age,String sex){
        this.name=name;
        this.age=age;
        this.sex=sex;
    }
    // 输出 JavaBean 的属性值方法
    public void printInfo(){
        System.out.println("用户姓名—"+name);             // 输出用户的姓名
        System.out.println("用户年龄—"+age);              // 输出用户的年龄
        System.out.println("用户性别—"+sex);              // 输出用户的性别
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    }
}
```

在 Spring 的配置文件 applicationContext.xml 中通过 <constructor-arg> 元素为 JavaBean 的属性赋值，关键代码如下：

```
<!-- User Bean -->
<bean name="user" class="com.mr.user.User">
   <constructor-arg>
          <value> 无语 </value>
   </constructor-arg>
   <constructor-arg>
          <value>30</value>
   </constructor-arg>
   <constructor-arg>
          <value> 女 </value>
   </constructor-arg>
</bean>
```

注意 容器通过多个 <constructor-arg> 标签为构造方法传递参数，如果标签的赋值顺序与构造方法中参数的顺序或类型不同，程序会产生异常，可以使用 <constructor-arg> 元素的“index”属性和“type”属性解决此类问题。

说明 index 属性用于指定当前 <constructor-arg> 标签为构造方法的哪个参数赋值，type 属性用于指定参数类型，当需要赋值的属性在构造方法中没有相同的类型时，可以使用这个参数。

创建类 Manager，在其 main() 方法中的关键代码如下：

```
Resource resource = new ClassPathResource("applicationContext.xml");
                                                    // 加载配置文件
BeanFactory factory = new XmlBeanFactory(resource);
User user = (User) factory.getBean("user");         // 获取 Bean
user.printInfo();                                   // 输出 JavaBean 的属性值
```

程序运行后控制台输出的信息如图 17.5 所示。

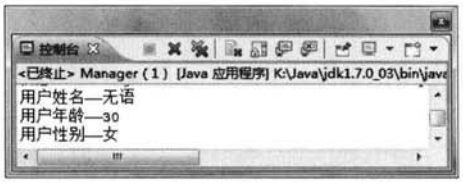


图 17.5 控制台输出的信息

技巧 由于存在大量的构造器参数，特别是当某些属性可选时可能使程序的效率低下。因此通常情况下，Spring 开发团队提倡使用 Setter 注入，这也是目前应用开发中最常使用的注入方式。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



技巧 构造器注入方式也有优点，它一次性将所有的依赖注入。在程序未完全初始化的状态下，注入对象不会被调用，此外对象也不可能再次被重新注入。对于注入类型的选择并没有硬性的规定，对于那些没有源代码的第三方类或者没有提供 Setter 方法的遗留代码，只能选择构造器注入方式实现依赖注入。

17.2.5　引用其他 Bean

Spring 利用 IoC 将 JavaBean 所需要的属性注入其中，不需要编写程序代码来初始化 JavaBean 的属性，使程序代码整洁且规范化。降低了 JavaBean 之间的耦合度，Spring 开发的项目中的 JavaBean 不需要修改任何代码即可应用到其他程序中，在 Spring 中可以通过配置文件使用 <ref> 元素引用其他 JavaBean 的实例对象。

【例 17.3】 将 User 对象注入到 Spring 的控制器 Manager 中，并在控制器中执行 User 的 printInfo() 方法。

实例位置：光盘 \MR\ 源码 \ 第 17 章 \17.3

在控制器 Manager 中注入 User 对象，关键代码如下：

```
public class Manager extends AbstractController {
    private User user;                                    // 注入 User 对象
    public User getUser() {
        return user;
    }
    public void setUser(User user) {
        this.user = user;
    }
protected ModelAndView handleRequestInternal(HttpServletRequest arg0,
            HttpServletResponse arg1) throws Exception {
        user.printInfo();                                 // 执行 User 中的信息打印方法
        return null;
    }
}
```

在上面的代码中，Manager 类继承自 AbstractController 控制器，该控制器是 Spring 中最基本的控制器，所有的 Spring 控制器都继承该控制器，它提供了诸如缓存支持和 mimetype 设置这样的功能。当一个类从 AbstractController 继承时，需要实现 handleRequestInternal() 抽象方法，该方法用来实现自己的逻辑，并返回一个 ModelAndView 对象，在本例中返回 null。

说明 如果在控制器中返回一个 ModelAndView 对象，那么该对象需要在 Spring 的配置文件 applicationContext.xml 中配置。

在 Spring 的配置文件 applicationContext.xml 中设置 JavaBean 的注入，关键代码如下：

```
<!-- 注入 JavaBean -->
<bean name="/main.do" class="com.mr.main.Manager">
   <property name="user">
         <ref local="user"/>
   </property>
</bean>
```

在 web.xml 文件中配置自动加载 applicationContext.xml 文件，在项目启动时 Spring 的配

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

置信息自动加载到程序中，所以在调用 JavaBean 时不再需要实例化 BeanFactory 对象：

```
<!-- 设置自动加载配置文件 -->
<servlet>
   <servlet-name>dispatcherServlet</servlet-name>
   <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
   class>
   <init-param>
          <param-name>contextConfigLocation</param-name>
          <param-value>/WEB-INF/applicationContext.xml</param-value>
   </init-param>
   <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
   <servlet-name>dispatcherServlet</servlet-name>
   <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

程序运行，在 IE 浏览器中单击“执行 JavaBean 的注入”超链接，在控制台将显示如图 17.6 所示的内容。



图 17.6　控制台输出的信息

17.2.6　创建匿名内部 JavaBean

在编程中经常遇到匿名的内部类，在 Spring 中需要匿名内部类的地方直接用 <bean> 标签定义一个内部类即可。如果要使这个匿名内部类，可以不指定 <bean> 标签的 id 或 name 属性，如下面这段代码：

```
<!-- 定义学生匿名内部类 -->
<bean id="school" class="School">
   <property name="student">
          <bean class="Student"/>
   </property>
</bean>
```

代码中定义了匿名的 Student 类，并将这个匿名内部类赋给了 School 类的实例对象。

17.3　AOP概述

Spring AOP 是继 Spring IoC 之后 Spring 框架的又一大特性，也是该框架的核心内容。AOP 是一种思想，所有符合该思想的技术都可以看做 AOP 的实现。Spring AOP 建立在 Java 的代理机制之上，Spring 框架已经基本实现了 AOP 的思想。在众多的 AOP 实现技术中，Spring AOP 做得最好，也是最为成熟的。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Spring AOP 的接口实现了 AOP 联盟（Alliance）定制标准化接口，这就意味着它已经走向了标准化，将得到更快的发展。

> **说明**　AOP 联盟由多个团体组成，这些团体致力于各个 Java AOP 子项目的开发。它们与 Spring 有相同的信念，即通过 AOP 使开发复杂的企业级应用变得更简单，且脉络更清晰。同时，他们也在为 AOP 制定标准化的统一接口，使得不同的 AOP 技术之间相互兼容。

17.3.1　AOP 术语

Spring AOP 实现基于 Java 的代理机制，从 JDK 1.3 开始就支持代理功能，但是性能却是一个很大的问题，为此出现了 CGLIB 代理机制。它可以生成字节码，所以其性能会高于 JDK 代理。Spring 支持这两种代理方式。但是随着 JVM（Java 虚拟机）性能的不断提高，这两种代理性能的差距越来越小。

Spring AOP 的有关术语如下。

（1）切面（aspect）

切面是对象操作过程中的截面，如图 17.7 所示。

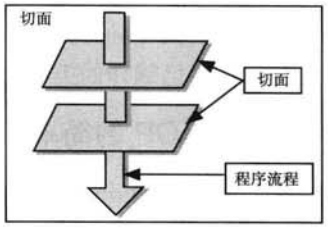


图 17.7　切面

如图所示，由于平行四边拦截了程序流程，所以 Spring 形象地将其称为“切面”。所谓的“面向切面编程”正是指如此，本书后面提到的“切面”即指这个“平行四边形”。

实际上“切面”是一段程序代码，这段代码将被“植入”到程序流程中。

（2）连接点（join point）

连接点指对象操作过程中的某个阶段点，如图 17.8 所示。

如图所示，在程序流程上的任意一点都可以是连接点。

它实际上是对象的一个操作，如对象调用某个方法、读写对象的实例或者某个方法抛出了异常等。

（3）切入点（pointcut）

切入点是连接点的集合，如图 17.9 所示。

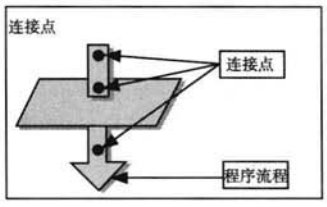


图 17.8　连接点

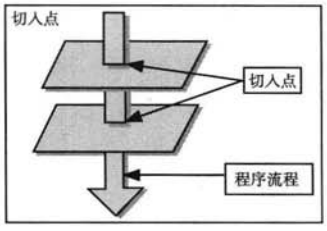


图 17.9　切入点

如图所示，切面与程序流程的“交叉点”即程序的切入点，确切地说，它是“切面注入”到程序中的位置，即“切面”是通过切入点被“注入”的。在程序中可以有多个切入点。

（4）通知（advice）

通知是某个切入点被横切后所采取的处理逻辑，即在“切入点”处拦截程序后通过通知

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

来执行切面，如图 17.10 所示。

（5）目标对象（target）

所有被通知的对象（也可以理解为被代理的对象）都是目标对象，目标对象及其属性改变、行为调用和方法传参的变化被 AOP 所关注，AOP 会注意目标对象的变动，并随时准备向目标对象“注入切面”。

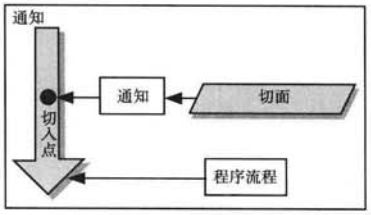


图 17.10　通知

（6）织入（weaving）

织入是将切面功能应用到目标对象的过程，有代理工厂并创建一个代理对象，这个代理可以为目标对象执行切面功能。

说明　AOP 的织入方式有 3 种，即编译时期（Compile time）织入、类加载时期（Classload time）织入和执行期（Runtime）织入。Spring AOP 一般多见于最后一种。

（7）引入（introduction）

对一个已编译的类（class），在运行时期动态地向其中加载属性和方法。

17.3.2　AOP 的简单实现

下例讲解 Spring AOP 简单实例的实现过程，以说明 AOP 编程的特点。

【例 17.4】　利用 Spring AOP 使日志输出与方法分离，在调用目标方法之前执行日志输出。

实例位置：光盘 \MR\ 源码 \ 第 17 章 \17.4

首先创建类 Target，它是被代理的目标对象。其中有一个 execute() 方法可以专注自己的职能，使用 AOP 对 execute() 方法输出日志，在执行该方法前输出日志。目标对象的代码如下：

```
public class Target {
    // 程序执行的方法
    public void execute(String name){
        System.out.println("程序开始执行：" + name);  // 输出信息
    }
}
```

通知可以拦截目标对象的 execute() 方法，并执行日志的输出，创建通知的代码如下：

```
public class LoggerExecute implements MethodInterceptor {
    public Object invoke(MethodInvocation invocation) throws Throwable {
        before();                    // 执行前置通知
    invocation.proceed();            //proceed() 方法是执行目标对象的 execute() 方法
        return null;
    }
    // 前置通知，before() 方法在 invocation.proceed() 之前执行，用于输出提示信息
  private void before() {
        System.out.println("程序开始执行！");
    }
}
```

使用 AOP 的功能必须创建代理，代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
public class Manger {
    //创建代理
    public static void main(String[] args) {
        Target target = new Target();                          //创建目标对象
        ProxyFactory di=new ProxyFactory();
        di.addAdvice(new LoggerExecute());
        di.setTarget(target);
        Target proxy=(Target)di.getProxy();
        proxy.execute(" AOP 的简单实现 ");                      //代理执行 execute() 方法
    }
}
```

程序运行后，在控制台输出的信息如图 17.11 所示。

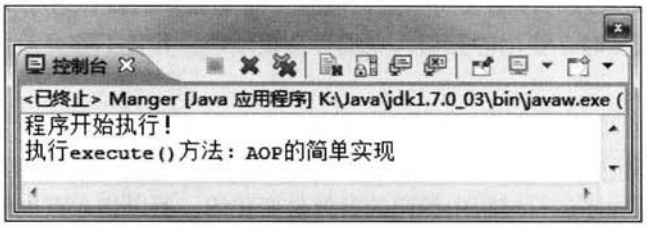


图 17.11　控制台输出的信息

17.4　Spring的切入点

Spring 的切入点（Pointcut）是 Spring AOP 比较重要的概念，它表示注入切面的位置。根据切入点织入的位置不同，Spring 提供了 3 种类型的切入点，即静态切入点、动态切入点和自定义切入点。

17.4.1　静态与动态切入点

静态与动态切入点需要在程序中选择使用。

（1）静态切入点

静态切入点可以为对象的方法签名，如在某个对象中调用了 execute() 方法时，这个方法即静态切入点。静态切入点需要在配置文件中指定，关键配置如下：

```
<bean id="pointcutAdvisor"
   class="org.springframework.aop.support.RegexpMethodPointcutAdvisor">
   <property name="advice">
          <ref bean="MyAdvisor" /><!-- 指定通知 -->
   </property>
   <property name="patterns">
          <list>
            <value>.*getConn*.</value><!-- 指定所有以 getConn 开头的方法名都是切入点 -->
                <value>.*closeConn*.</value>
          </list>
   </property>
</bean>
```

在上面的代码中，正则表达式 .*getConn*. 表示所有以 getConn 开头的方法都是切入点；

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

正则表达式 .*closeConn*. 表示所有以 closeConn 开头的方法都是切入点。

> **说明** 正则表达式由数学家 Stephen Kleene 于 1956 年提出，用其可以匹配一些指定的表达式，而不是列出每一个表达式的具体写法。

由于静态切入点只在代理创建时执行一次，然后缓存结果。下一次调用时直接从缓存中读取即可，所以在性能上要远高于动态切入点。第一次将静态切入点织入切面时，首先会计算切入点的位置，它通过反射在程序运行时获得调用的方法名。如果这个方法名是定义的切入点，则织入切面。然后缓存第一次计算结果，以后不需要再次计算，这样使用静态切入点的程序性能会高很多。

虽然使用静态切入点的性能会高一些，但是当需要通知的目标对象的类型多于一种，而且需要织入的方法很多时，使用静态切入点编程会很烦琐，并且不够灵活且会降低性能，这时可以选用动态切入点。

（2）动态切入点

静态切入点只能应用在相对不变的位置，而动态切入点可应用在相对变化的位置，如方法的参数上。由于在程序运行过程中传递的参数是变化的，所以切入点也随之变化，它会根据不同的参数来织入不同的切面。由于每次织入都要重新计算切入点的位置，而且结果不能缓存，所以动态切入点比静态切入点的性能要低得多。但是它能够随着程序中参数的变化而织入不同的切面，所以比静态切入点要灵活得多。

在程序中可以选择使用静态切入点和动态切入点，当程序对性能要求很高且相对注入不是很复杂时可以选用静态切入点；当程序对性能要求不是很高且注入比较复杂时可以使用动态切入点。

17.4.2 深入静态切入点

静态切入点在某个方法名上织入切面，所以在织入程序代码前要匹配方法名，即判断当前正在调用的方法是不是已经定义的静态切入点。如果是，说明方法匹配成功并织入切面；否则匹配失败，不织入切面。这个匹配过程由 Spring 自动实现，不需要编程的干预。

实际上 Spring 使用 boolean matches(Method, Class) 方法来匹配切入点，并利用 method.getName() 方法反射取得正在运行的方法名。在 boolean matches(Method,Class) 方法中，Method 是 java.lang.reflect.Method 类型；method.getName() 利用反射取得正在运行的方法名；Class 是目标对象的类型。该方法在 AOP 创建代理时被调用并返回结果，true 表示将切面织入，false 则不织入。静态切入点的匹配过程的代码如下：

```
<!-- 深入静态切入点 -->
<bean id=" pointcutAdvisor "
   class="org.springframework.aop.support.RegexpMethodPointcutAdvisor">
   <property name="patterns">
          <list>
                 <value>.*execute.*</value><!-- 指定切入点 -->
          </list>
   </property>
</bean>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Matches() 方法匹配成功后的代码如下：

```
public bollean matches(Method method,Class targetClass){
        return(method.getName().equals("execute"));              // 匹配切入点成功
}
```

17.4.3　深入切入点底层

掌握 Spring 切入点底层将有助于更加深刻的理解切入点。

Pointcut 接口是切入点的定义接口，用其来规定可切入的连接点的属性。通过扩展此接口可以处理其他类型的连接点，如域等（但是这样做很罕见）。定义切入点接口的代码如下：

```
public interface Pointcut {
   ClassFilter getClassFilter();
   MethodMatcher getMethodMatcher();
}
```

使用 ClassFilter 接口来匹配目标类，代码如下：

```
public interface ClassFilter {
   boolean matches(Class class);
}
```

可以看到，在 ClassFilter 接口中定义了 matches() 方法，即与……匹配。其中，class 代表被检测的 Class 实例，该实例是应用切入点的目标对象。如果返回 true，表示目标对象可以被应用切入点，否则不可以应用切入点。

使用 MethodMatcher 接口来匹配目标类的方法或方法的参数，代码如下：

```
public interface MethodMatcher {
   boolean matches(Method m,Class targetClass);
   boolean isRuntime();
   boolean matches(Method m,Class targetClass,Object[] args);
}
```

Spring 执行静态切入点还是动态切入点取决于 isRuntime() 方法的返回值。在匹配切入点之前 Spring 会调用 isRuntime() 方法，如果返回 false，则执行静态切入点，否则执行动态切入点。

17.4.4　Spring 中的其他切入点

Spring 提供了丰富的切入点供用户选择使用，目的是使切面灵活地注入到程序中的所需位置。例如，使用流程切入点可以根据当前调用堆栈中的类和方法来实施切入。Spring 常见的切入点如表 17.2 所示。

表 17.2　Spring 常见的切入点

切入点实现类	说　明
org.springframework.aop.support.JdkRegexpMethodPointcut	JDK 正则表达式方法切入点
org.springframework.aop.support.NameMatchMethodPointcut	名称匹配器方法切入点
org.springframework.aop.support.StaticMethodMatcherPointcut	静态方法匹配器切入点
org.springframework.aop.support.ControlFlowPointcut	流程切入点
org.springframework.aop.support.DynamicMethodMatcherPointcut	动态方法匹配器切入点

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

技巧 如果 Spring 提供的切入点无法满足开发需求，可以自定义切入点。Spring 提供的切入点很多，可以选择一个继承它并重载 matches 方法，也可以直接继承 Pointcut 接口并且重载 getClassFilter() 方法和 getMethodMatcher() 方法，这样可以编写切入点的实现。

17.5 Aspect对AOP的支持

Aspect 即 Spring 中所说的切面，它是对象操作过程中的截面，在 AOP 中是一个非常重要的概念。

17.5.1 Aspect 概述

Aspect 是对系统中的对象操作过程中截面逻辑进行模块化封装的 AOP 概念实体，通常情况下可以包含多个切入点和通知。

说明 AspectJ 是 Spring 框架 2.0 版本之后增加的新特性，Spring 使用了 AspectJ 提供的一个库来完成切入点的解析和匹配。但是 AOP 在运行时仍旧是纯粹的 Spring AOP，它并不依赖于 AspectJ 的编译器或者织入器，在底层中使用的仍然是 Spring 2.0 之前的实现体系。

使用 AspectJ 需要在应用程序的 classpath 中引入 org.springframework.aspects-3.1.1.RELEASE.jar，这个 jar 包可以在 Spring 的发布包的 dist 目录中找到。

例如，以 AspectJ 形式定义的 Aspect，代码如下：

```
aspect AjStyleAspect
{
    //切入点定义
    pointcut query(): call(public * get*(...));
    pointcut delete(): execution(public void delete(...));
    ...
    //通知
    before():query(){...}
    after returnint:delete(){...}
    ...
}
```

在 Spring 的 2.0 版本之后，可以通过使用 @AspectJ 的注解并结合 POJO 的方式来实现 Aspect。

17.5.2 Spring 中的 Aspect

最初在 Spring 中没有完全明确的 Aspect 概念，只是在 Spring 中的 Aspect 的实现和特性有所特殊而已，而 Advisor 就是 Spring 中的 Aspect。

Advisor 是切入点的配置器，它能将 Advice（通知）注入程序中的切入点的位置，并可以直接编程实现 Advisor，也可以通过 XML 来配置切入点和 Advisor。由于 Spring 的切入点具有多样性，而 Advisor 是为各种切入点设计的配置器，因此 Advisor 也有很多。

在 Spring 中的 Advisor 的实现体系由两个分支家族构成，即 PointctuAdvisor 和 Introdcution-Advisor 家族。家族的每个分支下都含有多个类和接口，其体系结构如图 17.12 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

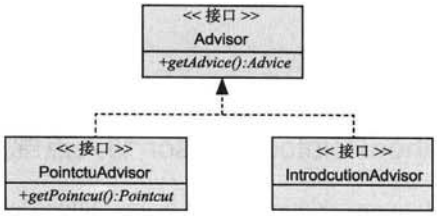


图 17.12　Advisor 的体系结构

在 Spring 中常用的两个 Advisor 都是 PointctuAdvisor 家族中的“子民”，它们是 DefaultPointcut Advisor 和 Name MatchMethodPointcutAdvisor。

17.5.3　DefaultPointcutAdvisor 切入点配置器

DefaultPointcutAdvisor 是位于 org.springframework.aop.support.DefaultPointcutAdvisor 包下的默认切入点通知者，它可以把一个通知配给一个切入点，使用之前首先要创建一个切入点和通知。

首先创建一个通知，这个通知可以自定义，关键代码如下：

```
public TestAdvice implements MethodInterceptor {
   public Object invoke(MethodInvocation mi) throws Throwable {
          Object Val=mi.proceed();
          return Val;
     }
}
```

然后创建自定义切入点，Spring 提供很多种类型的切入点，可以选择一个继承它并且分别重写 matches () 和 getClassFilter() 方法，以实现自己定义的切入点，关键代码如下：

```
public class TestStaticPointcut extends StaticMethodMatcherPointcut {
   public boolean matches (Method method Class targetClass){
                return ("targetMethod".equals(method.getName()));
   }
   public ClassFilter getClassFilter() {
                return new ClassFilter() {
                     public boolean matches(Class clazz) {
                        return (clazz==targetClass.class);
                }
                };
   }
}
```

分别创建一个通知和切入点的实例，关键代码如下：

```
Pointcut pointcut=new TestStaticPointcut ();                    //创建一个切入点
Advice advice=new TestAdvice ();                                //创建一个通知
```

如果使用 SpringAOP 的切面注入功能，需要创建 AOP 代理，通过 Spring 的代理工厂来实现，代码如下：

```
Target target =new Target();                                    //创建一个目标对象的实例
ProxyFactory proxy= new ProxyFactory();
proxy.setTarget(target);                                        //target 为目标对象
//前面已经对“advisor”做了配置，现在需要将“advisor”设置在代理工厂里
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
proxy.setAdivsor(advisor);
Target proxy = (Target) proxy.getProxy();
Proxy...// 此处省略的是代理调用目标对象的方法，目的是实施拦截注入通知
```

17.5.4 NameMatchMethodPointcutAdvisor 切入点配置器

此配置器位于 org.springframework.aop.support.NameMatchMethodPointcutAdvisor 包中，是方法名切入点通知者，使用它可以更加简洁地将方法名设置为切入点，关键代码如下：

```
NameMatchMethodPointcutAdvisor advice=new NameMatchMethodPointcutAdvisor(new
TestAdvice());
advice.addMethodName("targetMethod1name");
advice.addMethodName("targetMethod2name");
advice.addMethodName("targetMethod3name");
advice.addMethodName("targetMethod3name");
…// 可以继续添加方法的名称
…// 省略创建代理，可以参考上一节创建 AOP 代理内容
```

在上面的代码中，new TestAdvice() 为一个通知；advice.addMethodName("targetMethod1name") 方法的 targetMethod1name 参数是一个方法名称，advice.addMethodName("targetMethod1name") 表示将 targetMethod1name() 方法添加为切入点。

当程序调用 targetMethod1() 方法时会执行通知（TestAdvice）。

17.6 Spring的持久化

在 Spring 中关于数据持久化的服务主要是支持数据访问对象（DAO）和数据库 JDBC，其中数据访问对象是实际开发过程中应用比较广泛的技术。

17.6.1 DAO 模式

DAO（Data Access Object，数据访问对象）提供了读写数据库数据的一种方法，通过接口提供对外服务，程序的其他模块通过这些接口来访问数据库。这样会有很多好处，首先由于服务对象不再和特定的接口实现绑定在一起，使其易于测试。因为它提供的是一种服务，在不需要连接数据库的条件下即可进行单元测试，极大地提高了开发效率；其次通过使用与持久化技术无关的方法访问数据库，在应用程序的设计和使用上都有很大的灵活性，对于系统性能和应用也是一个飞跃。

说明 DAO 的主要作用是将持久性相关的问题与一般的业务规则和工作流隔离开来，它为定义业务层可以访问的持久性操作引入了一个接口并且隐藏了实现的具体细节。该接口的功能将依赖于采用的持久性技术而改变，但是 DAO 接口可以基本上保持不变。

DAO 属于 O/R Mapping 技术的一种，在该技术发布之前开发人员需要直接借助 JDBC 和 SQL 来完成与数据库的通信；在发布之后，开发人员能够使用 DAO 或其他不同的 DAO 框架来实现与 RDBMS（关系数据库管理系统）的交互。借助于 O/R Mapping 技术，开发人员能够将对象属性映射到数据表的字段并将对象映射到 RDBMS 中，这些 Mapping 技术能够为应用自动创建高效的 SQL 语句等；除此之外，O/R Mapping 技术还提供了延迟加载和缓存等高级

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

特征，而 DAO 是 O/R Mapping 技术的一种实现，因此使用 DAO 能够大量节省开发时间，并减少代码量和开发的成本。

17.6.2　Spring 的 DAO 理念

Spring 提供了一套抽象的 DAO 类供开发人员扩展，这有利于以统一的方式操作各种 DAO 技术，如 JDO 和 JDBC 等。这些抽象的 DAO 类提供了设置数据源及相关辅助信息的方法，而其中的一些方法与具体 DAO 技术相关。目前 Spring DAO 提供了如下抽象类。

- JdbcDaoSupport：JDBC DAO 抽象类，开发人员需要为其设置数据源（DataSource）。通过子类能够获得 JdbcTemplate 来访问数据库。
- HibernateDaoSupport：Hibernate DAO 抽象类，开发人员需要为其配置 Hibernate SessionFactory，通过其子类能够获得 Hibernate 实现。
- JdoDaoSupport：Spring 为 JDO 提供的 DAO 抽象类，开发人员需要为它配置 PersistenceManagerFactory，通过其子类能够获得 JdoTemplate。

在使用 Spring 的 DAO 框架存取数据库时，无须接触使用特定的数据库技术，通过一个数据存取接口来操作即可。

【例 17.5】 在 Spring 中利用 DAO 模式在 tb_user 表中添加数据。

实例位置：光盘 \MR\ 源码 \ 第 17 章 \17.5

该实例中 DAO 模式实现的示意如图 17.13 所示。

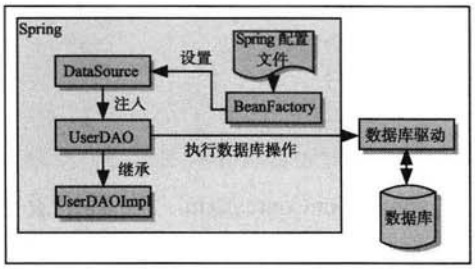


图 17.13　DAO 模式实现的示意

定义一个实体类对象 User，然后在类中定义对应数据表字段的属性，关键代码如下：

```
public class User {
    private Integer id;                          //唯一标识
    private String name;                         //姓名
    private Integer age;                         //年龄
    private String sex;                          //性别
    …                                            //省略的 Setter 和 Getter 方法
}
```

创建接口 UserDAOImpl，并定义用来执行数据添加的 insert() 方法。该方法使用的参数是 User 实体对象，代码如下：

```
public interface UserDAOImpl {
    public void inserUser(User user);                    //添加用户信息的方法
}
```

编写实现这个 DAO 接口的 UserDAO 类，并在其中实现接口中定义的方法。首先定义一

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

个用于操作数据库的数据源对象 DataSource，通过它创建一个数据库连接对象以建立与数据库的连接，这个数据源对象在 Spring 中提供了 javax.sql.DataSource 接口的实现，只需在 Spring 的配置文件中完成相关配置即可。这个类中实现了接口的抽象方法 insert()，通过这个方法访问数据库，关键代码如下：

```
public class UserDAO implements UserDAOImpl {
    private DataSource dataSource;                          //注入 DataSource
    public DataSource getDataSource() {
        return dataSource;
    }
    public void setDataSource(DataSource dataSource) {
        this.dataSource = dataSource;
    }
    //向数据表 tb_user 中添加数据
    public void inserUser(User user) {
        String name = user.getName();                       //获取姓名
        Integer age = user.getAge();                        //获取年龄
        String sex = user.getSex();                         //获取性别
        Connection conn = null;                             //定义 Connection
        Statement stmt = null;                              //定义 Statement
         try {
             conn = dataSource.getConnection();             //获取数据库连接
             stmt = conn.createStatement();
             stmt.execute("INSERT INTO tb_user (name,age,sex) "
                  + "VALUES('"+name+"','" + age + "','" + sex + "')");
                                                            //添加数据的 SQL 语句
         } catch (SQLException e) {
             e.printStackTrace();
         }
         …                                                  //省略的代码
}
```

编写 Spring 的配置文件 applicationContext.xml，在其中首先定义一个 JavaBean 名为“DataSource”的数据源，它是 Spring 中的 DriverManagerDataSource 类的实例。然后在配置前面编写 userDAO 类，并且注入其 DataSource 属性值，配置代码如下：

```
<!-- 配置数据源 -->
<bean id="dataSource" class="org.springframework.jdbc.datasource.
DriverManagerDataSource">
   <property name="driverClassName">
          <value>com.mysql.jdbc.Driver</value>
   </property>
   <property name="url">
          <value>jdbc:mysql://localhost:3306/db_database17</value>
   </property>
   <property name="username">
          <value>root</value>
   </property>
   <property name="password">
          <value>111</value>
   </property>
</bean>
<!-- 为 UserDAO 注入数据源 -->
<bean id="userDAO" class="com.mr.dao.UserDAO">
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    <property name="dataSource">
        <ref local="dataSource"/>
    </property>
</bean>
```

创建类 Manger，其 main() 方法中的关键代码如下：

```
Resource resource = new ClassPathResource("applicationContext.xml");
                                                    //加载配置文件
BeanFactory factory = new XmlBeanFactory(resource);
User user = new User();                             //实例化 User 对象
user.setName("张三");                               //设置姓名
user.setAge(new Integer(30));                       //设置年龄
user.setSex("男");                                  //设置性别
UserDAO userDAO = (UserDAO) factory.getBean("userDAO"); //获取 UserDAO
userDAO.inserUser(user);                            //执行添加方法
System.out.println("数据添加成功!!!");
```

运行程序后，数据表 tb_user 中添加的数据如图 17.14 所示。

	id	name	age	sex
▶	1	明日	30	男

图 17.14　tb_user 表中添加的数据

17.6.3　事务管理

Spring 中的事务基于 AOP 实现，而 Spring 的 AOP 以方法为单位，所以 Spring 的事务属性是对事务应用方法的策略描述。这些属性包括传播行为、隔离级别、只读和超时等属性。

说明　事务管理在应用程序中至关重要，它是一系列任务组成的工作单元，其中的所有任务必须同时执行。而且只有两种可能的执行结果，即全部成功和全部失败。

事务的管理通常分为如下两种方式。

1．编程式事务管理

在 Spring 中主要有两种编程式事务管理的实现方法，分别通过使用 Platform Transaction Manager 接口的事务管理器或 TransactionTemplate 实现。二者各有优缺点，推荐使用后者实现方式，因其符合 Spring 的模板模式。

说明　TransactionTemplate 模板和 Spring 的其他模板一样封装了打开和关闭资源等常用重复代码，在编写程序时只须完成需要的业务代码即可。

【例 17.6】　利用 TransactionTemplate 实现 Spring 编程式事务管理。

实例位置：光盘 \MR\ 源码 \ 第 17 章 \17.6

首先需要在 Spring 的配置文件中声明事务管理器和 TransactionTemplate，关键代码如下：

```
<!-- 定义 TransactionTemplate 模板 -->
<bean id="transactionTemplate" class="org.springframework.transaction.support.
TransactionTemplate">
    <property name="transactionManager">
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            <ref bean="transactionManager"/>
        </property>
        <property name="propagationBehaviorName">
        <!-- 限定事务的传播行为，规定当前方法必须运行在事务中，如果没有事务，则创建一个。一个新的事
        务和方法一同开始，随着方法的返回或抛出异常而终止 -->
            <value>PROPAGATION_REQUIRED</value>
        </property>
    </bean>
    <!-- 定义事务管理器 -->
    <bean id="transactionManager"
        class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource">
            <ref bean="dataSource"/>
        </property>
    </bean>
```

创建类 TransactionExample，定义添加数据的方法，在方法中执行两次添加数据库操作并用事务保护操作，关键代码如下：

```
public class TransactionExample {
    DataSource dataSource;                                  //注入数据源
    PlatformTransactionManager transactionManager;          //注入事务管理器
    TransactionTemplate transactionTemplate;                //注入 TransactionTemplate 模板
    …                                                       //省略的 Setter 和 Getter 方法
    public void transactionOperation() {
        transactionTemplate.execute(new TransactionCallback() {
            public Object doInTransaction(TransactionStatus status) {
                Connection conn = DataSourceUtils.getConnection(dataSource);
                                                            //获得数据库连接
                try {
                    Statement stmt = conn.createStatement();
                    //执行两次添加方法
                    stmt.execute("insert into tb_user(name,age,sex) values('小
                    强','26','男')");
                    stmt.execute("insert into tb_user(name,age,sex) values('小
                    红','22','女')");
                    System.out.println("操作执行成功！");
                } catch (Exception e) {
                    transactionManager.rollback(status);        //事务回滚
                    System.out.println("操作执行失败，事务回滚！");
                    System.out.println("原因："+e.getMessage());
                }
                return null;
            }
        });
    }
}
```

在上面的代码中，以匿名类的方式定义 TransactionCallback 接口的实现来处理事务管理。

创建类 Manger，其 main() 方法中的代码如下：

```
Resource resource = new ClassPathResource("applicationContext.xml");
                                                            // 加载配置文件
BeanFactory factory = new XmlBeanFactory(resource);
// 获取 TransactionExample
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
TransactionExample transactionExample = (TransactionExample) factory.getBean
("transactionExample");
transactionExample.transactionOperation();                    // 执行添加方法
```

为了测试事务是否配置正确，在 transactionOperation() 方法中执行两次添加操作的语句之间添加两句代码制造人为的异常。即当第 1 条操作语句执行成功后，第 2 条语句因为程序的异常无法执行成功。这种情况下如果事务成功回滚，说明事务配置成功，添加的代码如下：

```
int a=0;                                                      // 制造异常测试事务是否配置成功
a=9/a;
```

程序执行后控制台输出的信息如图 17.15 所示，数据表 tb_user 中没有插入数据。

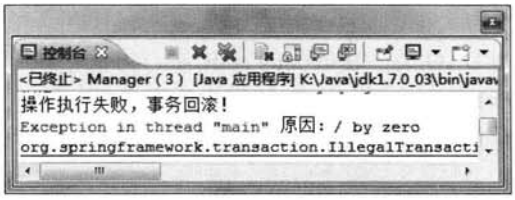


图 17.15　控制台输出的信息

2. 声明式事务管理

声明式事务不涉及组建依赖关系，它通过 AOP 实现事务管理，在使用声明式事务时不用编写任何代码即可实现基于容器的事务管理。Spring 提供了一些可供选择的辅助类，它们简化了传统的数据库操作流程。在一定程度上节省了工作量，提高了编码效率，所以推荐使用声明式事务管理。

在 Spring 中常用 TransactionProxyFactoryBean 完成声明式事务管理。

说明　使用 TransactionProxyFactoryBean 需要注入所依赖的事务管理器，并设置代理的目标对象、代理对象的生成方式和事务属性。代理对象是在目标对象上生成的包含事物和 AOP 切面的新对象，它可以赋给目标的引用来替代目标对象以支持事务或 AOP 提供的切面功能。

【例 17.7】 利用 TransactionProxyFactoryBean 实现 Spring 声明式事务管理。

实例位置：光盘 \MR\ 源码 \ 第 17 章 \17.7

在配置文件中定义数据源 DataSource 和事务管理器，该管理器被注入到 TransactionProxyFactoryBean 中，设置代理对象和事务属性。这里的目标对象的定义以内部类方式定义，配置文件中的关键代码如下：

```
<!-- 定义 TransactionProxy -->
<bean id="transactionProxy"
   class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean">
   <property name="transactionManager">
         <ref local="transactionManager" />
   </property>
   <property name="target">
               <!-- 以内部类的形式指定代理的目标对象 -->
         <bean id="addDAO" class="com.mr.dao.AddDAO">
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                        <property name="dataSource">
                            <ref local="dataSource" />
                        </property>
            </bean>
        </property>
        <property name="proxyTargetClass" value="true" />
        <property name="transactionAttributes">
            <props>
                <!-- 通过正则表达式匹配事务性方法，并指定方法的事务属性，即代理对象中只要是
                以 add 开头的方法名必须运行在事务中 -->
                <prop key="add*">PROPAGATION_REQUIRED</prop>
            </props>
        </property>
    </bean>
```

编写操作数据库的 AddDAO 类，在该类的 addUser() 方法中执行了两次数据插入操作。这个方法在配置 TransactionProxyFactoryBean 时被定义为事务性方法，并指定了事务属性，所以方法中的所有数据库操作都被当做一个事务处理。该类中的代码如下：

```
public class AddDAO extends JdbcDaoSupport {
    //添加用户的方法
    public void addUser(User user){
        //执行添加方法的 sql 语句
        String sql="insert into tb_user (name,age,sex) values('" +
                    user.getName() + "','" + user.getAge()+ "','" + user.
                    getSex()+ "')";
        //执行两次添加方法
        getJdbcTemplate().execute(sql);
        getJdbcTemplate().execute(sql);
    }
}
```

创建类 Manger，其 main() 方法中的代码如下：

```
Resource resource = new ClassPathResource("applicationContext.xml");
                                                                    //加载配置文件
BeanFactory factory = new XmlBeanFactory(resource);
AddDAO addDAO = (AddDAO)factory.getBean("transactionProxy");    //获取 AddDAO
User user = new User();                                          //实例化 User 实体对象
user.setName("张三");                                            //设置姓名
user.setAge(30);                                                 //设置年龄
user.setSex("男");                                               //设置性别
addDAO.addUser(user);                                            //执行数据库添加方法
```

可以延用例 17.6 中制造程序异常的方法测试配置的事务。

17.6.4 应用 JdbcTemplate 操作数据库

JdbcTemplate 类是 Spring 的核心类之一，可以在 org.springframework.jdbc.core 包中找到。该类在内部已经处理了数据库资源的建立和释放，并可以避免一些常见的错误，如关闭连接及抛出异常等，因此使用 JdbcTemplate 类简化了编写 JDBC 时所需的基础代码。

JdbcTemplate 类可以直接通过数据源的引用实例化，然后在服务中使用，也可以通过依赖注入的方式在 ApplicationContext 中产生并作为 JavaBean 的引用给服务使用。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明 JdbcTemplate 类运行了核心的 JDBC 工作流程，如应用程序要创建和执行 Statement 对象，只需在代码中提供 SQL 语句。该类可以执行 SQL 中的查询、更新或者调用存储过程等操作，并且生成结果集的迭代数据。它还可以捕捉 JDBC 的异常并转换为 org.springframework.dao 包中定义并能够提供更多信息的异常处理体系。

JdbcTemplate 类中提供了接口来方便访问和处理数据库中的数据，这些方法提供了基本的选项用于执行查询和更新数据库操作。JdbcTemplate 类提供了很多重载的方法用于数据查询和更新，提高了程序的灵活性。表 17.3 所示为 JdbcTemplate 中常用的数据查询方法。

表 17.3　JdbcTemplate 中常用的数据查询方法

方法名称	说　明
int QueryForInt(String sql)	返回查询的数量，通常是聚合函数数值
int QueryForInt(String sql,Object[] args)	
long QueryForLong(String sql)	返回查询的信息数量
long QueryForLong(String sql,Object[] args)	
Object queryforObject(string sql,Class requiredType)	返回满足条件的查询对象
Object queryforObject(string sql,Class requiredType,Object[] args)	
List queryForList(String sql)	返回满足条件的对象 List 集合
List queryForList(String sql,Object[] args)	

说明 sql 参数指定查询条件的语句，requiredType 指定返回对象的类型，args 指定查询语句的条件参数。

【例 17.8】 利用 JdbcTemplate 在数据表 tb_user 中添加用户信息。

实例位置：光盘 \MR\ 源码 \ 第 17 章 \17.8

在配置文件 applicationContext.xml 中配置 JdbcTemplate 和数据源，关键代码如下：

```
<!-- 配置 jdbcTemplate -->
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
   <property name="dataSource">
          <ref local="dataSource"/>
   </property>
</bean>
```

创建类 AddUser，获取 JdbcTemplate 对象，并利用其 update() 方法执行数据库的添加操作，其 main() 方法中的关键代码如下：

```
DriverManagerDataSource ds = null;
JdbcTemplate jtl = null;
Resource resource = new ClassPathResource("applicationContext.xml");// 获取配置文件
BeanFactory factory = new XmlBeanFactory(resource);
jtl =(JdbcTemplate)factory.getBean("jdbcTemplate");              // 获取 JdbcTemplate
String sql = "insert into tb_user(name,age,sex) values ('小明','23','男')";
                                                                 //SQL 语句
jtl.update(sql);                                                 // 执行添加操作
```

程序运行后，tb_user 表中添加的数据如图 17.16 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

id	name	age	sex
10	小明	23	男

图 17.16　tb_user 表中添加的数据

JdbcTemplate 类实现了很多方法的重载特征，在实例中使用了其写入数据的常用方法 update(String)。

17.6.5　与 Hibernate 整合

在 Spring 中整合 Hibernate 4 时，已经不再提供 HibenateTemplate 和 HibernateDaoSupport 类了，而只有一个称为 LocalSessionFactoryBean 的 SessionFactoryBean，通过它可以基于注解或是 XML 文件来配置映射文件。

Hibernate 的连接和事务管理等从建立 SessionFactory 类开始，该类在应用程序中通常只存在一个实例。因而其底层的 DataSource 可以使用 Spring 的 IoC 注入，之后注入 SessionFactory 到依赖的对象之中。

说明　*在应用的整个生命周期中只要保存一个 SessionFactory 实例即可。*

在 Spring 中配置 SessionFactory 对象，通过实例化 LocalSessionFactoryBean 类来完成，为了让该对象获取连接的后台数据库的信息，需要配置一个数据源 dataSource，配置方法如下：

```
<!-- 配置数据源 -->
<bean id="dataSource"
   class="org.springframework.jdbc.datasource.DriverManagerDataSource">
   <property name="driverClassName">
          <value>com.mysql.jdbc.Driver</value>
   </property>
   <property name="url">
          <value>jdbc:mysql://localhost:3306/db_database17
          </value>
   </property>
   <property name="username">
          <value>root</value>
   </property>
   <property name="password">
          <value>111</value>
   </property>
</bean>
```

通过一个 LocalSessionFactoryBean 配置 Hibernate，通过 Hibernate 的多个属性可以控制其行为。其中最重要的是 mappingResources 属性，通过其 value 值指定 Hibernate 使用的映射文件，代码如下：

```
<!-- 定义 Hibernate 的 sessionFactory -->
<bean id="sessionFactory" class="org.springframework.orm.hibernate4.
LocalSessionFactoryBean">
   <property name="dataSource">
   <ref bean="dataSource" />
   </property>
   <property name="hibernateProperties">
   <props>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    <!-- 数据库连接方言 -->
    <prop key="dialect">org.hibernate.dialect.SQLServerDialect</prop>
    <!-- 在控制台输出 SQL 语句 -->
    <prop key="hibernate.show_sql">true</prop>
    <!-- 格式化控制台输出的 SQL 语句 -->
    <prop key="hibernate.format_sql">true</prop>
    </props>
    </property>
    <!--Hibernate 映射文件 -->
    <property name=" mappingResources ">
    <list>
    <value>com/mr/User.hbm.xml</value>
    </list>
    </property>
</bean>
```

配置完成之后即可使用 Spring 提供的支持 Hibernate 的类，如通过 HibenateTemplate 类和 HibernateDaoSupport 子类可以实现 Hibernate 的大部分功能，为开发实际项目带来了方便。

17.6.6　整合 Spring 与 Hibernate

该实例主要演示在 Spring 中使用 Hibernate 框架完成数据持久化，它继承了 Spring 的 HibernateDaoSupport 类来创建操作数据的 UserDaoSupport 类，在其中编写完成数据库操作的方法。

【例 17.9】 整合 Spring 与 Hibernate，在 tb_user 表中添加信息。

实例位置：光盘 \MR\ 源码 \ 第 17 章 \17.9

首先创建 Spring 的配置文件 applicationContext.xml，用于配置数据源 datasource 和 LocalSessionFactoryBean。

编写一个执行数据库操作的 DAO 类文件 UserDAO，在该类中，首先定义一个 sessionFactory 属性，并为该属性添加对应的 Setter 和 Getter 方法，然后定义一个获取 Session 对象的方法 getSession()，最后再定义一个保存用户信息的方法，在该方法中调用 Session 对象的 save() 方法保存用户信息。UserDAO 类的关键代码如下：

```
public class UserDAO {
    private SessionFactory sessionFactory;                    // 定义 SessionFactory 属性
    // 保存用户的方法
    public void insert(User user) {
            this.getSession().save(user);
    }
    /**
     * 获取 Session 对象
     */
    protected Session getSession() {
            return sessionFactory.openSession();
    }
    public SessionFactory getSessionFactory() {
            return sessionFactory;
    }
    public void setSessionFactory(SessionFactory sessionFactory) {
            this.sessionFactory = sessionFactory;
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

将该类配置到 Spring 的配置文件中，同时为其 SessionFactory 属性注入 DataSource，代码如下：

```
<!-- 注入 SessionFactory -->
<bean id="userDAO" class="com.mr.dao.UserDAO">
   <property name="sessionFactory">
   <ref local="sessionFactory" />
   </property>
</bean>
```

创建类 AddUser，在其中调用添加用户的方法，其 main() 方法中的关键代码如下：

```
//添加用户信息
public static void main(String[] args) {
   Resource resource = new ClassPathResource("applicationContext.xml");
                                                          //获取配置文件
   BeanFactory factory = new XmlBeanFactory(resource);
   UserDAO userDAO = (UserDAO)factory.getBean("userDAO");  //获取 UserDAO
   User user = new User();                                 //实例化 User 对象
   user.setName("Spring 与 Hibernate 整合 ");               //设置姓名
   user.setAge(20);                                        //设置年龄
   user.setSex(" 男 ");                                    //设置性别
   userDAO.insert(user);                                   //执行用户添加的方法
   System.out.println(" 添加成功！ ");
}
```

程序运行后在 tb_user 表中添加的数据如图 17.17 所示。

id	name	age	sex
12	Spring与Hibernate整合	20	男

图 17.17　tb_user 表中添加的数据

17.7　上机实践

17.7.1　登录页面国际化

➤➤➤ 题目描述

对于一个大型的网站，实现页面国际化很重要，这样可以适应不同国家浏览者的需要。本上机实践尝试开发一个程序，应用 Spring 实现登录页面国际化，运行结果如图 17.18 所示。



图 17.18　运行结果

➤➤➤ 技术指导

在 Spring 中定义了 MessageSource 接口来解析消息资源，由于 ApplicationContext 容器实现了 MessageSource 接口，所以它可以解析国际化资源。

消息的国际化需要编写多个对应特定语言的资源文件和一个必需的默认资源文档，如消息的资源名为 messages，那么对应的默认资源文件是 messages.properties，它是容器找不到对应语言资源时默认读取的资源文件。messages_en_US.properties 对应美国英语资源，其他语言如德语和墨西哥语言等都有对应的资源文件。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

17.7.2　利用 Spring 整合 Hibernate 来批量添加数据

➤➤➤ 题目描述

本上机实践尝试编写一个程序，利用 Spring 整合 Hibernate 来实现批量用户注册，运行后在 tb_user 表中批量添加的数据如图 17.19 所示。

	id	name	business	addTime
▶	1	明日科技0	高级工程师	2012-08-04 11:18:06
	2	明日科技1	高级工程师	2012-08-04 11:18:06
	3	明日科技2	高级工程师	2012-08-04 11:18:06
	4	明日科技3	高级工程师	2012-08-04 11:18:06
	5	明日科技4	高级工程师	2012-08-04 11:18:07

图 17.19　在 tb_user 表中批量添加的数据

➤➤➤ 技术指导

首先定义用户信息的实体对象，JavaBean 中封装的属性包括 id 值、用户名称、职务及添加时间；然后创建 DAOSupport 类，封装批量添加的方法；并创建 Sping 的控制器，在其中注入封装批量添加方法的类，最后在 web.xml 中配置自动加载 Spring 的配置文件。

17.7.3　利用 AOP 实现用户注册

➤➤➤ 题目描述

在本章介绍了 Spring 的 AOP 技术，本上机实践尝试应用 AOP 编程实现用户注册的功能。主要目的是让读者通过简单的操作来理解 AOP 的思想，运行效果如图 17.20 所示。在用户注册页面中输入用户注册信息，单击“注册”按钮，在控制台上输出执行结果。

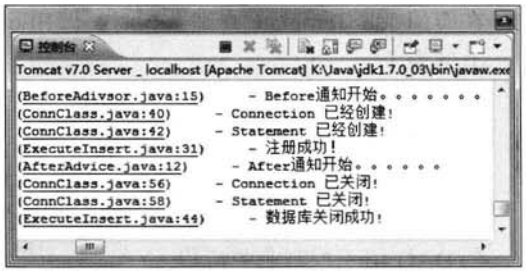


图 17.20　在控制台上输出的结果

➤➤➤ 技术指导

在实现本实例时，主要应用 AOP 中的 Before 通知和 After 通知来实现。Before 通知与 After 通知的区别如下。

Before 通知会在目标对象的方法执行之前执行。在具体应用中需要实现 org.springframework.aop.MethodBeforeAdvice 接口，并且要重写默认的 before() 方法，该方法会在目标对象所指定的方法之前执行。在 before() 方法执行完后，如果没有异常，将会接着执行目标对象的方法。

After 通知与 Before 通知非常相似，不过它会在目标对象的方法执行之后执行。在具体应用中实现 org.springframework.aop.AfterReturningAdvice 接口，还需要实现该接口中的 afterReturning() 方法，该方法会在目标对象所指定的方法之后执行。

17.8　小结

本章详细讲解了 Spring 核心技术 IoC、Bean 的基础知识、AOP 和 Spring 持久化。其中，核心技术 IoC 是 Spring 框架的基础；Bean 可为读者理解 IoC 技术奠定坚实基础；AOP 技术提供了另一种编程思想，即面向切面编程；Spring 持久化支持数据访问对象（DAO）和数据库 JDBC，其中数据访问对象是实际开发过程中应用比较广泛的技术。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

17.9 习题

选择题

1. 以下为 Spring 的 applocationContext.xml 中配置 bean 依赖注入的相关代码，假设需要将 TestB 的 bean 对象注入到 TestA 中。以下选项中，在 TestA 的 <bean> 元素中添加的代码正确的是（　　）。

```
<bean id="testB" class="com.lh.TestB ">
<bean id="testA" class="com.lh.TestA ">

</bean>
```

A. <property name="testA" ><ref local="testB"></property>
B. <property name="testB" ><ref local="testA"></property>
C. <property name="testB" ><ref local="testB"></property>
D. 以上选项都不对

2. 在应用 JdbcTemplate 查询数据时，调用 queryForList(String sql) 方法返回的 List 集合的元素类型为（　　）。
A. List 类型　　B. String 类型　　C. Map 类型　　D. Set 类

3. 关于 DAO，下列说法错误的是（　　）。
A. DAO 代表数据访问对象（Data Access Object）
B. DAO 属于 O/R Mapping 技术的一种
C. DAO 用于控制业务流程
D. DAO 用于操作数据库

4. Spring 中的核心类有哪些（　　）。（多选）
A. BeanFactory：产生一个新的实例，可以实现单例模式
B. BeanWrapper：提供统一的 get 及 set 方法
C. ApplicationContext:：提供框架的实现，包括 BeanFactory 的所有功能
D. SessionFactory ：一个 SessionFactory 实例对应一个数据存储源，从 SessionFactory 中获取 Session

填空题

1. IoC 是指_____。
2. AOP 是指_____。
3. 依赖注入的两种方法包括_____和_____。
4. Spring 的事务管理的两种方式分别是_____和_____。
5. Spring 的 AOP 提供的三种切入点分别是_____、_____和_____。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第四篇
项目实战篇

本篇主要内容：

- 第 18 章　基于 Ajax 的聊天室
- 第 19 章　基于 SSH2 的网络商城

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 18 章　基于 Ajax 的聊天室

视频讲解：37 分钟

聊天室通常是交友、沟通、技术交流等活动的桥梁。通常情况下，聊天室具有显示在线人员列表、发送并显示实时聊天信息等功能。为了达到实时显示的目的，在实现聊天室时，通常使用 Ajax 技术。本章将介绍一个基于 Ajax 实现的无刷新聊天室。

通过学习本章，读者可以达到以下学习目的：

- 熟悉聊天室的开发流程；
- 掌握编写并配置字符编码过滤器的方法；
- 掌握实时显示在线人员列表的方法；
- 掌握实时显示聊天信息的方法；
- 掌握退出聊天室的两种方法。

18.1　需求分析

随着互联网的飞速发展，聊天室这种比较传统的交流方式已经被众人所认可。通过聊天室在线聊天已成为网络上人与人之间沟通、交流和联系的一种方式。为此，越来越多的网站开始提供在线聊天的功能。与此同时，聊天室也以其方便、快捷、低成本等优势受到众多企业的青睐，很多企业的网站中也加入了聊天室，以达到增进企业与消费者之间、消费者与消费者之间相互交流和联系的目的。

18.2　系统设计

18.2.1　系统目标

根据对网络中的聊天室的需求进行分析，制定目标如下：

- 操作简单方便、界面友好美观。
- 具有滚屏功能。
- 具有悄悄话功能。
- 能够保存聊天记录。
- 系统运行稳定、安全可靠。

18.2.2　系统功能结构

聊天室的基本功能就是聊天，其他功能都是对基本功能的强化。本章中的聊天室主要包括以下功能。

- 用户登录，包括验证用户是否在线。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 实时显示在线人员列表及在线人数。
- 用户发言，包括私聊、显示聊天内容和滚屏显示聊天信息。
- 安全退出聊天室，包括单击“关闭”按钮 × 退出聊天室及将长时间不发言者踢出聊天室。

18.2.3　系统流程图

聊天室的系统流程如图 18.1 所示。

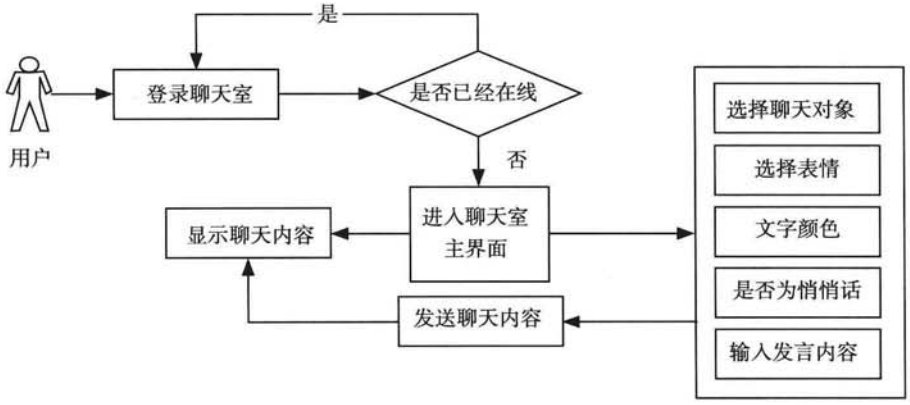


图 18.1　聊天室的系统流程图

18.3　系统开发及运行环境

碧蓝晴空聊天室的开发环境如下。

（1）服务器端

- 操作系统：Windows 7、Windows XP 或者 Windows Server 2003。
- Web 服务器：Tomcat 7.0。
- Java 开发包：JDK 1.7 以上。
- 开发工具：Eclipse IDE for Java EE。
- 浏览器：IE 8.0 或者更高版本。
- 显示器分辨率：最佳效果 1024 像素 ×768 像素。

（2）客户端

- 浏览器：IE 8.0 或者更高版本。
- 分辨率：最佳效果 1024 像素 ×768 像素。

18.4　系统文件夹组织结构

在编写项目代码之前，需要制定好项目的系统文件夹组织结构，这样不但可以保证团队开发的一致性，也可以规范系统的整体架构。创建完系统中可能用到的文件夹或者 Java 包之后，在开发时，只需将创建的类文件或者资源文件保存到相应的文件夹中即可。碧蓝晴空聊天室的文件夹组织结构如图 18.2 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

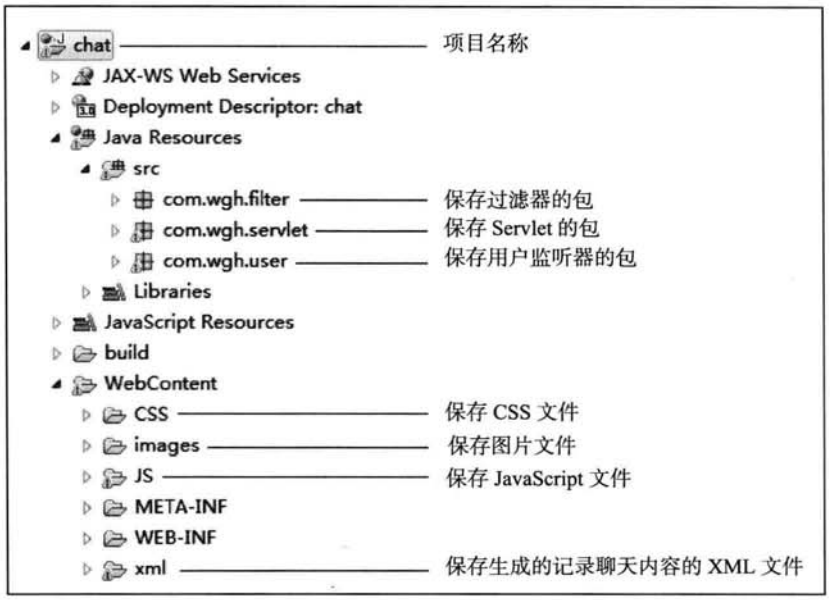


图 18.2　文件夹组织结构

18.5　公共类设计

公共类是代码重用的一种形式，它将各个功能模块经常调用的方法提取到公用的 Java 类中。使用公共类，不但实现了项目代码的重用，还提高了程序的性能和代码的可读性。本节将介绍碧蓝晴空聊天室中的公共类设计。

18.5.1　编写并配置字符编码过滤器

Step 1) 创建字符编码过滤器对象，其名称为 CharactorFilter 类。此类实现继承自 javax.servlet.Filter 接口，并在 doFilter() 方法中对请求中的字符编码格式进行设置，其关键代码如下：

```
/**
 * Servlet 过滤器实现类 CharactorFilter
 */
public class CharactorFilter implements Filter {
   String encoding = null;                                    // 字符编码
    public CharactorFilter() {
    }
   /**
    * 销毁方法
    */
   public void destroy() {
          encoding = null;
   }
   /**
    * 过滤处理方法
    */
   public void doFilter(ServletRequest request, ServletResponse response,
   FilterChain chain)
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
throws IOException, ServletException {
        if(encoding != null){                                    // 判断字符编码是否为空
            request.setCharacterEncoding(encoding);// 设置请求的编码格式
            // 设置 response 字符编码
                response.setContentType("text/html;charset="+encoding);
        }
        chain.doFilter(request, response);                       // 传递给下一个过滤器
    }
    /**
     * 初始化方法
     */
    public void init(FilterConfig fConfig) throws ServletException {
        encoding = fConfig.getInitParameter("encoding");         // 获取初始化参数
    }
}
```

CharactorFilter 类是实例中的字符编码过滤器，它主要通过在 doFilter() 方法中指定 request 与 reponse 两个参数的字符集 encoding 来进行编码处理，使得目标资源的字符集支持中文。其中，encoding 是 CharactorFilter 类定义的字符编码格式成员变量，此变量在过滤器的初始化方法 init() 中被赋值，它的值是通过 FilterConfig 对象读取配置文件中的初始化参数获取的。

注意　在过滤器对象的 doFilter() 方法中，业务逻辑处理完成之后，需要通过 FilterChain 对象的 doFilter() 方法将请求传递到下一个过滤器或目标资源，否则将出现错误。

Step 2 在创建了过滤器对象之后，还需要对过滤器进行一定的配置才可以正常使用，采用注解的方式配置过滤器 CharactorFilter 的代码如下：

```
@WebFilter(
        urlPatterns = { "/*" },
        initParams = {
                @WebInitParam(name = "encoding", value = "UTF-8")
        })                                                       // 配置过滤器
```

在过滤器 CharactorFilter 的配置声明中，将它的初始化参数 encoding 的值设置为 UTF-8，它与 JSP 页面的编码格式相同，支持中文。

注意　在配置过滤器时，URL 映射可以使用正则表达式进行配置，如本实例中使用“/*”来匹配所有请求。

18.5.2　编写聊天室相关的 Servlet 实现类

本模块是采用 JSP+JavaBean+Servlet 模式开发的，Servlet 主要完成对页面的逻辑跳转工作。聊天室相关请求的 Servlet 实现类继承了 HttpServlet 类，它的主要方法是 doGet() 和 doPost() 方法，这两个方法会被自动执行。方法本身没有具体的事务，它是根据 HttpServletRequest 类的 getParameter() 方法获取 method 参数值来执行相应方法的。

聊天室相关请求的 Servlet 实现类的关键代码如下：

```
package com.wgh.servlet;
import java.io.*;                                    // 导入 java.io 包中的全部类文件
import java.text.*;                                  // 导入 java.text 包中的全部类文件
import java.util.*;                                  // 导入 java.util 包中的全部类文件
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
import javax.servlet.annotation.WebServlet;          // 导入 servlet 注解
import javax.servlet.http.HttpServlet;               // 导入 HttpServlet 类
import javax.servlet.http.HttpServletRequest;        // 导入 HttpServletRequest 类
import javax.servlet.http.HttpServletResponse;       // 导入 HttpServletResponse 类
import javax.servlet.http.HttpSession;               // 导入 HttpSession 类
import javax.servlet.http.HttpSessionBindingEvent;
                                                     // 导入 HttpSessionBindingEvent 类
import org.dom4j.Document;                           // 导入 dom4j 包中的 Document 类
import org.dom4j.Element;                            // 导入 dom4j 包中的 Element 类
import org.dom4j.io.OutputFormat;                    // 导入 dom4j 包中的 OutputFormat 类
import org.dom4j.io.SAXReader;                       // 导入 dom4j 包中的 SAXReader 类
import org.dom4j.io.XMLWriter;                       // 导入 dom4j 包中的 XMLWriter 类
import com.wgh.user.UserInfo;                        // 导入对在线用户具体操作的类
import com.wgh.user.UserListener;                    // 导入用户监听器
@WebServlet("/MessagesAction")                       // 配置过滤器
public class MessagesAction extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response) {
        doPost(request, response);                   // 调用 doPost() 方法
    }
    public void doPost(HttpServletRequest request, HttpServletResponse response) {
        String action = request.getParameter("action");
            if ("getMessages".equals(action)) { // 从 XML 文件中读取聊天信息
                this.getMessages(request, response);
            } else if ("sendMessage".equals(action)) {
                                                     // 发送聊天信息
                this.sendMessages(request, response);
            } else if ("loginRoom".equals(action)) {
                                                     // 登录时，写入系统公告
                this.loginRoom(request, response);
            } else if ("exitRoom".equals(action)) {
                  this.exitRoom(request, response);
                                                     // 退出时，写入系统公告
            }
    }
    …                          // 此处省略了该类中的其他方法，这些方法将在后面的具体过程中给出
}
```

18.6 用户登录模块设计

当用户访问聊天室时，首先进入的是登录聊天室页面。在该页面中，用户输入自己喜欢的用户名，如图 18.3 所示。单击 GO 按钮，系统将判断输入的用户名是否被占用，如果被占用将给出提示，否则进入系统主界面。



图 18.3 用户登录页面

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

实现登录聊天室的具体步骤如下。

Step 1) 设计登录聊天室的页面，该页面主要用于收集用户输入的登录信息，以及通过自定义的 JavaScript 函数验证输入信息是否为空和是否包括非法字符，该页面的表单元素只包括输入用户名的文本框和提交表单的按钮，具体代码如下：

```
<form name="form1" method="post" action="login.jsp" onSubmit="return check()">
    用户名：<input type="text" name="username">
    <input type="image" name="imageField" src="images/go.jpg">
</form>
```

Step 2) 在聊天室的 Servlet 实现类中，添加登录聊天室的方法 loginRoom()，在该方法中，首先获取登录用户，然后判断该用户是否登录，如果已经登录，给出提示信息，并将该用户添加到在线用户列表中，同时向保存聊天信息的 XML 文件中添加一条系统公告信息，最后将页面重定向到登录成功页。loginRoom() 方法的具体代码如下：

```
public void loginRoom(HttpServletRequest request,HttpServletResponse response) {
        response.setContentType("text/html;charset=UTF-8");
        HttpSession session = request.getSession();
        String username = request.getParameter("username");
                                                    //获得登录用户名
        UserInfo user = UserInfo.getInstance();     //获得 UserInfo 类的对象
        session.setMaxInactiveInterval(600);  //设置 Session 的过期时间为 10 分钟
        Vector vector = user.getList();
        boolean flag = true;                        //标记是否登录的变量
        // 判断用户是否登录
        if (vector != null && vector.size() > 0) {
            if (vector.contains(username)) {
                PrintWriter out;
                try {
                    out = response.getWriter();
                    out.println("<script language='javascript'>
                    alert('该用户已经登录');window.location.href=
                    'index.jsp';</script>");
                } catch (IOException e) {
                    e.printStackTrace();
                }
                flag = false;
            }
        }
        // 保存用户信息
        if (flag) {
            UserListener ul = new UserListener(); //创建 UserListener 的对象
            ul.setUser(username);                 //添加用户
            user.addUser(ul.getUser());       //添加用户到 UserInfo 类的对象中
            session.setAttribute("user", ul);
            //将 UserListener 对象绑定到 Session 中
            session.setAttribute("username", username);
                                                  //保存当前登录的用户名
            session.setAttribute("loginTime", new Date().toLocaleString());
                                                  // 保存登录时间
            /** ****************** 开始系统公告 ****************************
            ***** */
            String newTime = new SimpleDateFormat("yyyyMMdd").format(new
            Date());
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                String fileURL = request.getSession().getServletContext().
                getRealPath("xml/" + newTime + ".xml");
                createFile(fileURL);    //判断 XML 文件是否存在，如果不存在则创建该文件
                try {
                        SAXReader reader = new SAXReader();
                                //  实例化 SAXReader 对象
                        Document feedDoc = reader.read(new File(fileURL));
                                //获取 XML 文件对应的 XML 文档对象
                        Element root = feedDoc.getRootElement();
                                //获取根节点
                        Element messages = root.element("messages");
                                //获取 messages 节点
                        Element message = messages.addElement("message");
                                //创建子节点 message
                        message.addElement("from").setText("[ 系统公告 ]");
                                //创建子节点 from
                        message.addElement("face").setText("");
                                //创建子节点 face
                        message.addElement("to").setText("");
                                //创建子节点 to
                        message.addElement("content").addCDATA(
                                "<font color='gray'>" + username + "走进
                                了聊天室！ </font>");// 创建子节点 content
                        message.addElement("sendTime").setText(new Date().
                        toLocaleString()); //创建子节点 sendTime
                        message.addElement("isPrivate").setText("false");
                                // 创建子节点
                        request.getRequestDispatcher("login_ok.jsp").forward
                        (request,response);
                        OutputFormat format = new OutputFormat();
                                //创建 OutputFormat 对象
                        XMLWriter writer = new XMLWriter(new FileWriter
                        (fileURL),format);
                        writer.write(feedDoc); // 向流写入数据
                        writer.close(); // 关闭 XMLWriter
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```

Step 3 编写登录成功页 login_ok.jsp，在该文件中将页面重定向到聊天室的主界面。登录成功页面的具体代码如下：

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<% response.sendRedirect("main.jsp");%>
```

当用户成功登录到聊天室后，系统将向保存聊天信息的 XML 文件中写入一条公告信息。例如，当“琦琦”登录后，将在保存聊天信息的 XML 文件中添加以下代码：

```
<message>
  <from>[ 系统公告 ]</from>
  <face />
  <to />
  <content>&lt;font color='gray'&gt; 琦琦走进了聊天室！ &lt;/font&gt;</content>
  <sendTime>2012-8-6 14:26:41</sendTime>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    <isPrivate>false</isPrivate>
  </message>
```

说明　保存聊天信息的 XML 文件以当天的系统日期命名。例如，2012 年 8 月 6 日的聊天信息将被保存在 20120806.xml 文件中。

18.7　聊天室主体功能模块设计

聊天室的主体功能模块主要包括实时获取并显示在线人员列表、实现用户发言、实时显示聊天内容等。其中，在实现用户发言时，可以有两种形式，一种是所有人都可以看到的信息，另一种是私聊，也就是发送只有发言人和接收人可以看到的信息。

18.7.1　设计聊天室主页面

在用户登录页面中输入用户名，单击 GO 按钮即可进入聊天室的主界面。在聊天室的主页面中包括页面头部、在线人员列表区、聊天内容显示区和用户发言区 4 部分，具体布局如图 18.4 所示。

<table>
<tr><td colspan="2"><table>
页面头部
</table></td></tr>
<tr><td><td id="online">
在线人员列表区
</td></td><td><td>

聊天内容显示区

</td></td></tr>
<tr><td colspan="2"><table>
用户发言区
</table></td></tr>
</table>

图 18.4　主界面布局图

聊天室主界面主要用采用 <table> 标记进行布局，其关键代码如下：

```
…        <!-- 此处省略了页面头部的 HTML 代码 -->
<table width="778" height="276" border="0" align="center" cellpadding="0"
cellspacing="0">
  <tr>
    <td width="165" valign="top" bgcolor="#F7FDED" id="online"
    style="padding:5px">
    在线人员列表
    </td>
    <td width="613"  height="200px" valign="top" bgcolor="#FFFFFF"
    style="padding:5px; ">
    <div style="height:290px; overflow:hidden" id="content">聊天内容</div>
    </td>
  </tr>
</table>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<table width="778" border="0" align="center" cellpadding="0" cellspacing="0"
bordercolor="#D6D3CE" bgcolor= "#D1E9AA">
…        <!-- 此处省略了用户发言区的代码，该代码将在第 18.7.3 小节进行详细介绍 -->
</table>
```

18.7.2 实时获取并显示在线人员列表

在开发聊天室程序时，为了让用户及时了解在线用户，并与其他用户进行交流，需要提供实时获取并显示在线人员列表的功能。本章介绍如何在聊天室的主界面的左侧显示在线人员列表及在线人数，其效果如图 18.5 所示。



图 18.5 在线人员列表

在实现实时显示在线人员列表时，应用了 Ajax 技术，这里我们编写了用于进行 Ajax 重构的 AjaxRequest.js 文件，对于该文件的具体代码请参见光盘。实现实时显示在线人员列表的具体步骤如下。

Step 1 编写自定义的 JavaScript 函数 showOnline()，用于实例化 Ajax 对象。showOnline() 函数的具体代码如下：

```
function showOnline(){
    var loader=new net.AjaxRequest("online.jsp?nocache="+new Date().
    getTime(),deal_online,onerror,"GET");
}
```

在上面的代码中，一定要有代码“?nocache="+new Date().getTime()”，否则将出现在线人员列表不更新的情况。

Step 2 从上面的代码中可以看出，Ajax 进行异步请求目标的 URL 地址为 online.jsp，即 JSP 文件。在该文件中，主要是将保存在集合类中的在线人员列表显示到页面。online.jsp 页面的代码如下：

```
<%@ page contentType="text/html; charset=UTF-8" language="java" import="java.
util.*" %>
<%@ page import="com.wgh.UserInfo"%>
<% request.setCharacterEncoding("UTF-8"); %>
<%
UserInfo list=UserInfo.getInstance();
Vector vector=list.getList();                                    //获取在线人员列表
int amount=0;
%>
<table width="100%" border="0" cellpadding="0" cellspacing="0">
  <tr><td height="32" align="center" class="word_orange ">欢迎来到碧蓝晴空聊天室! </
  td></tr>
  <tr>
  <td height="23" align="center"><a  href="#" onclick="set('所有人')">所有人</
  a></td>
  </tr>
 <%if(vector!=null&&vector.size()>0){
    String username="";
    amount=vector.size();
        for(int i=0;i<amount;i++){
            username=(String)vector.elementAt(i);          //获取在线用户的用户名
        %>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    <tr>
      <td height="23" align="center">                    <!-- 显示在线用户的用户名 -->
          <a href="#" onclick="set('<%=username%>')"><%=username%></a></td>
    </tr>
  <%}%>
  <tr><td height="30" align="center">当前在线 [<font color="#FF6600"><%=amount%></
  font>] 人 </td></tr>     <!-- 显示在线人数 -->
  </table>
```

Step 3 在聊天室的主界面中，将左侧用于显示在线人员列表的单元格的 id 属性设置为 online，用于实时显示在线人员列表，具体代码如下：

```
<td width="165" valign="top" bgcolor="#F7FDED" id="online" style="padding:5px">在
线人员列表 </td>
```

Step 4 编写 Ajax 的回调函数 deal_online()，用于将获取的在线人员列表赋值给 id 为 online 的 <td> 标记的 innerHTML 属性。deal_online() 函数的具体代码如下：

```
function deal_online(){
    online.innerHTML=this.req.responseText;
}
```

Step 5 为了让页面载入后就调用 Ajax 获取在线人员列表，并且每隔 10 秒便获取一次数据，还需要在页面中添加 JavaScript 代码，具体代码如下：

```
window.setInterval("showOnline();",10000);
window.onload=function(){
    showOnline();                          // 当页面载入后显示在线人员列表
}
```

18.7.3　实现用户发言

聊天室程序中一个必不可少的功能就是实现用户发言，在本章介绍的聊天室中，用户发言区位于聊天室主界面的底部，如图 18.6 所示。



图 18.6　用户发言区

实现用户发言的具体步骤如下。

Step 1 在页面的合适位置添加用于收集用户发言信息的表单及表单元素，关键代码如下：

```
<form action="" name="form1" method="post" >
    <input name="from" type="hidden" value="<%=session.
    getAttribute("username")%>"> [<%=session. GetAttribute ("username")%> ] 对 <input
    name="to" type="text" value="" size="35" readonly="readonly">
表情
<select name="face" class="wenbenkuang">
  <option  value=" 无表情的 "> 无表情的 </option>
  <option value=" 微笑着 " selected> 微笑着 </option>
  <option value=" 笑呵呵地 "> 笑呵呵地 </option>
  <!-- 引处省略了添加其他列表项的代码 -->
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
  <option value=" 无精打采的 "> 无精打采的 </option>
</select>
说：  悄悄话
<input name="isPrivate" type="checkbox" class="noborder" id="isPrivate"
value="true" onClick="checkIsPrivate()">
滚屏
<input name="scrollScreen" type="checkbox" class="noborder" id="scrollScreen"
onClick="checkScrollScreen()" value= "1" checked>
    字体颜色：
      <select name="color" size="1" class="wenbenkuang" id="select">
        <option selected> 默认颜色 </option>
        <option style="color:#FF0000" value="FF0000"> 红色热情 </option>
        <option style="color:#0000FF" value="0000FF"> 蓝色开朗 </option>
        <!-- 引处省略了添加其他列表项的代码 -->
        <option style="color:#999999" value="999999"> 烟雨蒙蒙 </option>
      </select>
    <input name="content1" type="text" size="70" onKeyDown="if(event.keyCode==13
    && event.ctrlKey) {send();}">
    <input name="Submit2" type="button" class="btn_blank" value=" 发送 "
    onClick="send()">
    <input name="button_exit" type="button" class="btn_orange" value=" 退出聊天室 "
    onClick="Exit()">
</form>
```

在上面的代码中，语句 <%=session.getAttribute("username")%> 用于显示当前的登录用户名。

聊天对象文本框被设置为只读属性时用户就不能手动输入聊天对象，因此还需要提供选择聊天对象的功能，这可以通过在主页面中添加选择聊天对象的 JavaScript 自定义函数及在线人员列表中添加超级链接实现。实现将选择的聊天对象添加到聊天对象文本框的 JavaScript 代码如下：

```
function set(selectPerson){                          // 自动添加聊天对象
    if(selectPerson!="<%=session.getAttribute("username")%>"){
        if(form1.isPrivate.checked && selectPerson==" 所有人 "){
            alert(" 请选择私聊对象！ ");
        }else{
            form1.to.value=selectPerson;
        }
    }else{
        alert(" 请重新选择聊天对象！ ");
    }
}
```

说明 关于为在线人员列表添加超链接的代码可以参见第 18.7.2 小节。

Step 2 编写自定义的 JavaScript 函数 send()，用于调用 Ajax 实现用户发言。在该函数中，首先验证用户输入信息的合法性，然后再将提交的表单元素的内容连接为一个参数字符串，最后实例化 Ajax 对象。send() 函数的具体代码如下：

```
function send(){          // 验证聊天信息并发送
    if(form1.to.value==""){
        alert(" 请选择聊天对象！ ");return false;
    }
    if(form1.content1.value==""){
        alert(" 发送信息不可以为空！ ");form1.content1.focus();return false;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    }
    if(form1.isPrivate.checked){          // 根据私聊复选框的值设置变量 isPrivate 的值
        isPrivate="true";
    }else{
        isPrivate="false";
    }
    var param="from="+form1.from.value+"&face="+form1.face.value+"&color="+form1.
    color.value+"&to="+form1.to.value+"&content="+form1.content1.
    value+"&isPrivate="+isPrivate;
    var loader=new net.AjaxRequest("MessagesAction?action=sendMessage",deal_
    send,onerror,"POST",param);
}
```

Step 3 在聊天室相关的 Servlet 实现类中添加发送聊天信息的方法 sendMessages()。在该方法中，首先获取用户发言的相关信息，并对出现的中文信息进行转码，然后判断保存当天聊天信息的 XML 文件是否存在，如果不存在则创建该文件，最后将聊天信息保存到 XML 文件中，并重定向网页。sendMessages() 方法的具体代码如下：

```
public void sendMessages(HttpServletRequest request,HttpServletResponse
response) {
        response.setContentType("text/html;charset=UTF-8");
        Random random = new Random();
        String from = request.getParameter("from");              // 发言人
        String face = request.getParameter("face");              // 表情
        String to = request.getParameter("to");                  // 接收者
        String color = request.getParameter("color");            // 字体颜色
        String content = request.getParameter("content");        // 发言内容
        String isPrivate = request.getParameter("isPrivate"); // 是否为悄悄话
        String sendTime = new Date().toLocaleString();           // 发言时间
        /** ****************** 开始添加聊天信息 ********************************** */
        String newTime = new SimpleDateFormat("yyyyMMdd").format(new Date());
        String fileURL = request.getSession().getServletContext().
        getRealPath("xml/" + newTime + ".xml");
        createFile(fileURL);              // 判断文件是否存在，当文件不存在时创建该文件
        try {
                SAXReader reader = new SAXReader();
                                  // 实例化 SAXReader 对象
                Document feedDoc = reader.read(new File(fileURL));
                                  // 获取 XML 文件对应的 XML 文档对象
                Element root = feedDoc.getRootElement();
                                  // 获取根节点
                Element messages = root.element("messages");
                                  // 获取 messages 节点
                Element message = messages.addElement("message");
                                  // 创建子节点 message
                message.addElement("from").setText(from);
                                  // 创建子节点 from
                message.addElement("face").setText(face);
                                  // 创建子节点 face
                message.addElement("to").setText(to);
                                  // 创建子节点 to
                message.addElement("content").addCDATA(
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                                "<font color='" + color + "'>" + content + "</
                                font>");// 创建子节点 content
                message.addElement("sendTime").setText(sendTime);
                                                // 创建子节点 sendTime
                message.addElement("isPrivate").setText(isPrivate);
                                                // 创建子节点
                OutputFormat format = new OutputFormat();
                                                // 创建 OutputFormat 对象
                request.getRequestDispatcher(
                "MessagesAction?action=getMessages&nocache="+ random.nextInt(10000)).
                forward(request, response);
                XMLWriter writer = new XMLWriter(new FileWriter(fileURL),
                format);
                writer.write(feedDoc);          // 向流写入数据
                writer.close();                 // 关闭 XMLWriter
                /** ***************** 添加结束 ******************************** */
        } catch (Exception e) {
                e.printStackTrace();
        }
}
```

在上面的代码中调用了 createFile() 方法，该方法用于判断指定的文件（即以当前日期命名的保存聊天信息的 XML 文件）是否存在，如果不存在则创建该文件，具体代码如下：

```
public void createFile(String fileURL) {
        /** ************** 判断 XML 文件是否存在，如果不存在则创建该文件 ********** */
        File file = new File(fileURL);
        if (!file.exists()) {           // 判断文件是否存在，如果不存在，则创建该文件
                try {
                        file.createNewFile();   // 创建文件
                        String dataStr = "<?xml version=\"1.0\" encoding=\"UTF-
                        8\"?>\r\n";
                        dataStr = dataStr + "<chat>\r\n";
                        dataStr = dataStr + "<messages></messages>";
                        dataStr = dataStr + "</chat>\r\n";
                        byte[] content = dataStr.getBytes();
                        FileOutputStream fout = new FileOutputStream(file);
                        fout.write(content);    // 将数据写入输出流
                        fout.flush();           // 刷新缓冲区
                        fout.close();           // 关闭输出流
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }
}
```

18.7.4 实时显示聊天内容

聊天室程序中另一个必不可少的功能就是实时显示聊天内容。在本章介绍的聊天室中，聊天内容显示区位于聊天室主界面的右侧，如图 18.7 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 18.7　聊天内容显示区

实现显示聊天内容应用了 Ajax 技术，这里我们编写了用于进行 Ajax 重构的 AjaxRequest.js 文件，该文件的具体代码请参见光盘。实现实时显示聊天内容的具体步骤如下。

Step 1 编写自定义的 JavaScript 函数 showContent() 用于实例化 Ajax 对象。showContent() 函数的具体代码如下：

```
function showContent(){
    var loader1=new net.AjaxRequest("MessagesAction?action=getMessages&nocache="
    +new Date().getTime(), deal_ content,onerror,"GET");
}
```

Step 2 从上面的代码中可以看出，Ajax 进行异步请求目标的 URL 地址为 MessagesAction?action=getMessages。从该 URL 地址可以看出，在进入主界面前会调用聊天室相关的 Servlet 实现类中的 getMessages() 方法。在该方法中，首先判断保存当前聊天信息的 XML 文件是否存在，如果不存在则创建它，然后开始解析保存聊天信息的 XML 文件，并将聊天内容连接为一个字符串，最后将连接后的字符串保存到 HttpServletRequest 对象中，并重定向网页到输入聊天内容的页面 content.jsp。getMessages() 方法的具体代码如下：

```
public void getMessages(HttpServletRequest request,HttpServletResponse response) {
    response.setContentType("text/html;charset=UTF-8");
    String newTime = new SimpleDateFormat("yyyyMMdd").format(new Date());
    String fileURL = request.getSession().getServletContext().
    getRealPath("xml/" + newTime + ".xml");
    createFile(fileURL);                                    // 当文件不存在时创建该文件
    /****************** 开始解析保存聊天内容的 XML 文件 ***********************/
    try {
          SAXReader reader = new SAXReader();     // 实例化 SAXReader 对象
          Document feedDoc = reader.read(new File(fileURL));
                                                  // 获取 XML 文件对应的 XML 文档对象
          Element root = feedDoc.getRootElement();        // 获取根节点
          Element channel = root.element("messages");     // 获取 messages 节点
          Iterator items = channel.elementIterator("message");
                                                          // 获取 message 节点
          String messages = "";
          // 获取当前用户
          HttpSession session = request.getSession();
          String userName = "";
          if (null == session.getAttribute("username")) {
                request.setAttribute("messages", "error");
                                                  // 保存标记信息，表示用户账户已经过期
          } else {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                userName = session.getAttribute("username").toString();
                DateFormat df = DateFormat.getDateTimeInstance();
                while (items.hasNext()) {
                        Element item = (Element) items.next();
                        String sendTime = item.elementText("sendTime");
                                        //获取发言时间
                        try {
                              if (df.parse(sendTime).after(df.parse
                              (session.getAttribute("loginTime").toString()))
                                    || sendTime.equals
                              (session.getAttribute("loginTime").toString())) {
                                 String from = item.elementText("from");
                                        //获取发言人
                                 String face = item.elementText("face");
                                        //获取表情
                                 String to = item.elementText("to");
                                        //获取接收者
                                 String content = item.elementText("content");
                                        //获取发言内容
                                 boolean isPrivate = Boolean.valueOf(item.
                                 elementText("isPrivate"));
                                 if (isPrivate) {// 获取私聊内容
                                       if (userName.equals(to)||
                                       userName.equals(from)) {
                                          messages += "<font color='red'>
                                          [私人对话]</font><font color='blue'><b>"
                                          + from+ "</b></font><font
                                          color='#CC0000'>"+ face
                                          + "</font>对<font color='green'>["+
                                          to+ "]</font>说："
                                          + content+ " <font color='gray'>["
                                          + sendTime + "]</font><br>";
                                       }
                                 } else if ("[系统公告]".equals(from)) {
                                                // 获取系统公告信息
                                       messages += "[系统公告]：" + content+
                                       " <font color='gray'>["
                                                + sendTime + "]</font><br>";
                                 } else {       // 获取普通发言信息
                                       messages += "<font color='blue'><b>" + from
                                                + "</b></font><font
                                       color='#CC0000'>"
                                                + face+ "</font>对<font
                                       color='green'>[" + to
                                                + "]</font>说：" + content+
                                       " <font color='gray'>["
                                                + sendTime + "]</font><br>";
                                 }
                              }
                        } catch (Exception e) {
                              System.out.println("" + e.getMessage());
                        }
                }
                request.setAttribute("messages", messages);  //保存获取的聊天信息
        }
        request.getRequestDispatcher("content.jsp").forward(request,response);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Step 3 编写显示聊天内容的 JSP 页面，在该页面中只需要应用 out 对象的 println() 方法将返回的执行结果输出即可，具体代码如下：

```
<%@ page contentType="text/html; charset=UTF-8" language="java" import="java.
util.*" errorPage="" %>
<% request.setCharacterEncoding("UTF-8"); %>
<%out.println(request.getAttribute("messages").toString());
%>
```

Step 4 在聊天室的主界面中，在右侧显示聊天内容的单元格中添加一个 id 属性为 content 的 标记，用于实时显示聊天内容，具体代码如下：

```
<div style="height:290px; overflow:hidden" id="content"> 聊天内容 </div>
```

Step 5 编写 Ajax 的回调函数 deal_content()，在该函数中，首先获取 Ajax 处理页的返回值，然后去除字符串中的 Unicode 空白符，最后判断在获取信息时是否产生错误，如果是则退出聊天室，否则将获取的聊天内容赋值给 id 为 content 的 标记的 innerHTML 属性。deal_content() 函数的具体代码如下：

```
function deal_content(){
    var returnValue=this.req.responseText;        // 获取 Ajax 处理页的返回值
    var h=returnValue.replace(^s/g,"");           // 去除字符串中的 Unicode 空白符
    if(h=="error"){                  // 判断在获取信息时是否产生错误，如果是则退出聊天室
        Exit();
    }else{
        content.innerHTML=sysBBS+returnValue+"</span>"; // 显示聊天内容
    }
}
```

Step 6 为了让页面载入后就调用 Ajax 获取聊天内容，并且每隔 1 秒便获取一次数据，还需要在页面中添加 JavaScript 代码，具体代码如下：

```
window.setInterval("showContent();",1000);
window.onload=function(){
    showContent();                                    // 当页面载入后显示聊天内容
}
```

18.8　退出聊天室模块

在该聊天室中提供了两种退出聊天室的方法，一种是单击主页面中的“退出聊天室”按钮，另一种是单击浏览器的“关闭”按钮 。需要注意的是，无论采用哪种方法都会弹出图 18.8 所示的对话框。



图 18.8　退出聊天室时显示的对话框

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

18.8.1 实现安全退出聊天室功能

安全退出聊天室是指单击“退出聊天室”按钮时退出聊天室。实现安全退出聊天室的具体步骤如下。

Step 1) 在主页面的合适位置添加“退出聊天室”按钮，并在按钮的 onclick 事件中调用自定义的 JavaScript 函数 Exit()，关键代码如下：

```
<input name="button_exit" type="button" class="btn_orange" value="退出聊天室"
onClick="Exit()">
```

Step 2) 编写自定义的 JavaScript 函数 Exit()，在该函数中首先将页面重定向到退出聊天室页面 leave.jsp，然后再弹出“欢迎您下次光临！”对话框，具体代码如下：

```
function Exit(){
    window.location.href="MessagesAction?action=exitRoom";
    alert("欢迎您下次光临！");
}
```

Step 3) 从上面的代码中可以看出，Ajax 进行异步请求目标的 URL 地址为 MessagesAction?action=exitRoom。从该 URL 地址可以看出，退出聊天室时，会调用聊天室相关的 Servlet 实现类中的 exitRoom() 方法。在该方法中，首先获取 HttpSession，然后销毁 session，最后重定向网页到用户登录页面 index.jsp。exitRoom() 方法的具体代码如下：

```
public void exitRoom(HttpServletRequest request,HttpServletResponse response) {
    HttpSession session = request.getSession();          // 获取 HttpSession
    session.invalidate();                                // 销毁 session
    try {
        request.getRequestDispatcher("index.jsp").forward(request, response);
                                                         // 重定向页面到登录页面
    } catch (Exception e) {
        e.printStackTrace();                             //输出异常信息到控制台
    }
}
```

Step 4) 当用户退出聊天室时，为了让其他用户及时了解到该信息，还需要向保存聊天内容的 XML 文件中写入公告信息，以显示给其他用户。在实现该功能时，首先需要在 MessagesAction.java 文件中编写 writeInfo() 方法，在该方法中向保存聊天内容的 XML 文件写入用户离开聊天室的公告信息，具体代码如下：

```
public void writeInfo(String user, HttpSessionBindingEvent arg0) {
        HttpSession session = arg0.getSession();      //获取 HttpSession
        String newTime = new SimpleDateFormat("yyyyMMdd").format(new Date());
        String fileURL = session.getServletContext().getRealPath(
                    "xml/" + newTime + ".xml");
                    //获取保存聊天内容的 XML 文件的文件名
                    createFile(fileURL);
                    //判断 XML 文件是否存在，如果不存在则创建该文件
        // 获取当前用户
        try {
                SAXReader reader = new SAXReader();   // 实例化 SAXReader 对象
                Document feedDoc = reader.read(new File(fileURL));
                                                  //获取 XML 文件对应的 XML 文档对象
                Element root = feedDoc.getRootElement();
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                                                    //获取根节点
            Element messages = root.element("messages");
                                                    // 获取 messages 节点
            Element message = messages.addElement("message");
                                                    // 创建子节点 message
            message.addElement("from").setText("[ 系统公告 ]");
                                                    // 创建子节点 from
            message.addElement("face").setText("");// 创建子节点 face
            message.addElement("to").setText(""); // 创建子节点 to
            message.addElement("content").addCDATA(
                        "<font color='gray'>" + user + " 离开聊天室! </
                        font>");                    // 创建子节点 content
            message.addElement("sendTime").setText(new Date().
            toLocaleString());                      // 创建子节点 sendTime
            message.addElement("isPrivate").setText("false");
                                                    // 创建子节点
            OutputFormat format = new OutputFormat();
                                                    // 创建 OutputFormat 对象
            XMLWriter writer = new XMLWriter(new FileWriter(fileURL),
            format);
            writer.write(feedDoc);                  // 向流写入数据
            writer.close();                         // 关闭 XMLWriter
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Step 5) 接下来还需要在监听用户状态的监听器 UserListener 的 valueUnbound() 方法（当 session 有对象移除时执行的方法）中，实例化 MessagesAction 的对象 message，并调用 MessagesAction 类的 writeInfo() 方法以将用户下线信息写入系统公告，关键代码如下：

```
MessagesAction message=new MessagesAction();    // 实例化 MessagesAction 的对象
message.writeInfo(this.user,arg0);              // 将用户下线信息写入系统公告
```

例如，当“琦琦”退出聊天室时，将向保存聊天内容的 XML 文件写入以下公告信息。

```
<message>
  <from>[ 系统公告 ]</from>
  <face />
  <to />
  <content>&lt;font color='gray'&gt; 琦琦离开聊天室! &lt;/font&gt;</content>
  <sendTime>2012-6-8 15:16:41</sendTime>
  <isPrivate>false</isPrivate>
</message>
```

18.8.2　处理非正常退出聊天室

非正常退出聊天室通常是指单击浏览器的“关闭”按钮退出聊天室。在实现单击“关闭”按钮退出聊天室时，只需要在聊天室的主界面中添加以下代码即可。

```
<script language="jscript">
window.onbeforeunload=function(){   // 当用户单击浏览器中的 " 关闭 " 按钮时，执行退出操作
   ver = navigator.appVersion;      // 浏览器的版本
   bType =navigator.appName;        // 浏览器的类型
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    vNumber=parseFloat(ver.substring(ver.indexOf("MSIE")+5,ver.
    lastIndexOf("Windows")));
    if(bType=="Microsoft Internet Explorer"){
          if(vNumber>6){                          //IE 6 以上浏览器
                vOffset=document.body.offsetWidth-document.body.scrollWidth;
                if(event.clientY<0 && event.clientX>document.body.scrollWidth-
                vOffset-20){
                      Exit();          //执行退出操作
                }else{                           //IE 6 及以下浏览器
                      if(event.clientY<0 && event.clientX>document.body.
                      scrollWidth){
                            Exit();          //执行退出操作
                      }
                }
          }
    }
}</script>
```

18.9 开发常见问题与解决方法

问题描述：在保存聊天信息时，由于组合后的聊天信息中包含 < 和 > 等特殊字符，为了让其正常显示，就需要将它们转换为对应的实体引用。例如，下面这条信息就是 dom4j 自动进行实体引用后的内结果。

```
<content>&lt;font color=''&gt; 你好，有时间吗？ &lt;/font&gt;</content>
```

从这段代码中，可以看到转换后的 XML 文档的可读性比较差。如何能让这些特殊字符正常显示，但不进行实体引用呢?

解决方法：在向 XML 添加内容节点时，不能使用 addText() 方法，而需要使用 addCDATA() 方法，具体的代码如下：

```
message.addElement("content").addCDATA("<font color='" + color + "'>" + content +
"</font>");   // 创建子节点 content
```

18.10 小结

本章重点讲解了碧蓝晴空聊天室系统中关键模块的开发过程。通过对本章的学习，读者应该能够熟悉 Ajax 聊天室的开发流程，并重点掌握如何编写过滤器以实现实时获取并显示在线人员列表，以及应用 Ajax 无刷新显示信息。另外，读者还应该掌握如何应用 dom4j 组件保存聊天信息到 XML 文件中。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 19 章　基于 SSH2 的网络商城

视频讲解：49 分钟

喜欢网上购物的读者一定登录过淘宝，也一定被网页上显示的琳琅满目的商品所吸引，从而忍不住拍一个自己喜爱的商品。如今已有越来越多的人加入到网购的行列，或成为网上店铺的老板，或成为新时代的购物潮人。你是否也想过开发一个自己的网上商城？在接下来的章节中我们将一起进入网络商城开发的旅程。

通过学习本章，读者可以达到以下学习目的：

- 熟悉网络商城的开发流程；
- 掌握 SSH2 的搭建流程；
- 掌握公共类的设计；
- 了解如何发布网站；
- 了解如何配置 DNS 服务器；
- 掌握应用 Struts 2 时如何解决中文乱码问题。

19.1　需求分析

近年来，随着 Internet 的迅速发展，互联网用户的爆炸式增长以及互联网对传统行业的冲击让其成为人们快速获取、发布和传递信息的重要渠道，于是电子商务逐渐流行起来，越来越多的商家在网上建起网上商城，向消费者展示出一种全新的购物理念，同时也有越来的越多的网友加入到了网上购物的行列。

笔者充分利用 Internet 这个平台，实现了这种全新的购物方式——网上购物，其目的是方便广大网友购物，让网友足不出户就可以逛商城买商品，为此构建“GO 购网络商城”。

19.2　系统设计

19.2.1　系统目标

GO 购网络商城系统是基于 B/S 模式的电子商务网站，用于满足不同人群的购物需求，笔者通过对现有的商务网站的考察和研究，从经营者和消费者的角度出发，以高效管理、满足消费者需求为原则，确定本系统满足以下要求：

- 统一友好的操作界面，具有良好的用户体验；
- 商品分类详尽，可按不同类别查看商品信息；
- 推荐产品、人气商品以及热销产品的展示；
- 会员信息的注册及验证；
- 用户可通过关键字搜索指定的产品信息；

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- ❑ 用户可通过购物车一次购买多件商品；
- ❑ 实现收银台的功能，用户选择商品后可以在线提交订单；
- ❑ 提供简单的安全模型，用户必须先登录，才允许购买商品；
- ❑ 用户可查看自己的订单信息；
- ❑ 设计网站后台，管理网站的各项基本数据；
- ❑ 系统运行安全稳定、响应及时。

19.2.2 系统功能结构

GO 购网络商城系统分为前台和后台两个部分，前台的台功能结构如图 19.1 所示。后台的功能结构如图 19.2 所示。

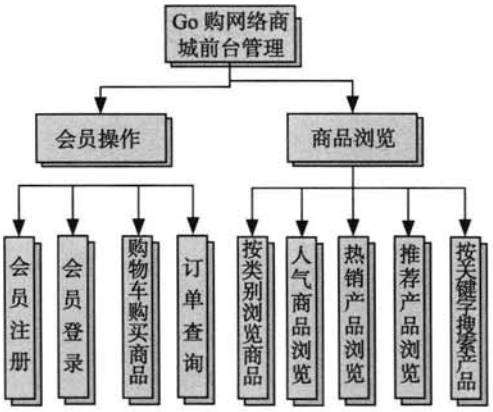


图 19.1 GO 购网络商城前台的功能结构

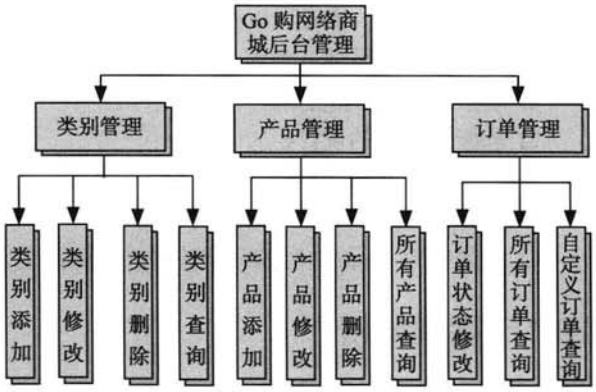


图 19.2 GO 购网络商城系统后台的功能结构

19.2.3 系统业务流程图

GO 购网络商城系统的系统业务流程图如图 19.3 所示。

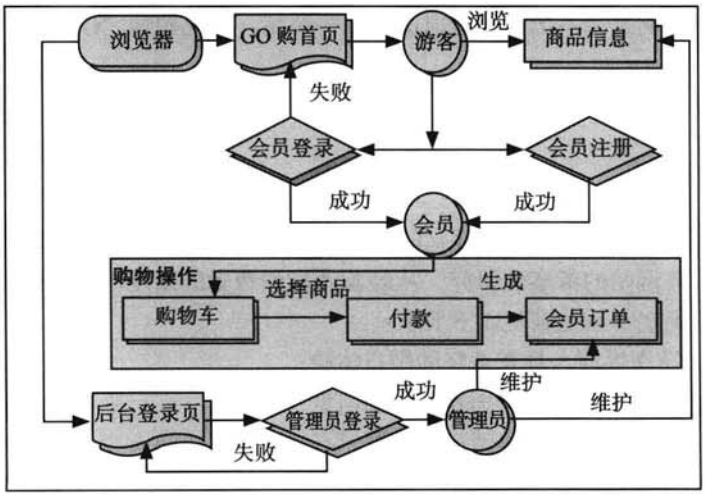


图 19.3 GO 购网络商城系统业务流程图

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

19.2.4　系统编码规范

开发应用程序常常需要团队合作来完成，每个人负责不同的业务模块，为了使程序的结构与代码风格统一标准化，增加代码的可读性，需要在编码之前制定一套统一的编码规范。下面介绍 GO 购网络商城开发中的编码规范。

（1）数据库

数据库以“db_”开头，后面接数据库相关英文单词或缩写进行命名，如表 19.1 所示。

表 19.1　数据库命名

数据库名称	描　述
db_shop	GO 购网络商城数据库

（2）数据表

数据表以“tb_”开头（小写），后面加数据表相关英文单词或缩写。下面将举例进行说明，如表 19.2 所示。

表 19.2　数据表命名

数据表名称	描　述
tb_order	订单信息表

（3）字段

字段统一采用英文单词或词组（可利用翻译软件）命名，如找不到专业的英文单词或词组，可以用相同意义的英文单词或词组代替。下面将举例进行说明，如表 19.3 所示。

表 19.3　字段命名

字段名称	描　述
orderId	订单编号
address	联系地址

说明　*在数据库中使用命名规范，有助于其他用户更好地理解数据表及表中各字段的内容。*

19.3　系统开发及运行环境

GO 购网络商城的开发环境如下。

（1）服务器端

- 操作系统：Windows 7、Windows XP 或者 Windows Server 2003。
- Web 服务器：Tomcat 7.0。
- Java 开发包：JDK 1.7 以上。
- 开发工具：Eclipse IDE for Java EE。
- 数据库：MySQL 5.1。
- 浏览器：IE 8.0 或者更高版本。
- 显示器分辨率：最佳效果 1024 像素 ×768 像素。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（2）客户端

- 浏览器：IE 8.0 或者更高版本。
- 分辨率：最佳效果 1024 像素 ×768 像素。

19.4 数据库与数据表设计

开发应用程序时，对数据库的操作是必不可少的，数据库是根据程序的需求及其实现功能所制定的，数据库设计的合理性将直接影响到程序的性能。本系统采用 MySQL 数据库，通过 Hibernate 实现系统的持久化操作。

19.4.1 E-R 图设计

为核心实体对象设计的 E-R 图如下。

1）tb_customer（会员信息表）的 E-R 图如图 19.4 所示。

2）表的 tb_order（订单信息表）的 E-R 图如图 19.5 所示。

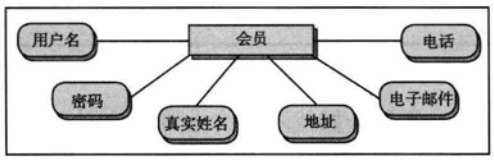


图 19.4 tb_customer 表的 E-R 图

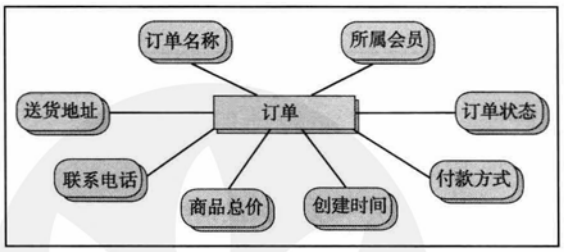


图 19.5 tb_order 表的 E-R 图

3）tb_orderitem（订单条目信息表）的 E-R 图如图 19.6 所示。

4）tb_productinfo（商品信息表）的 E-R 图如图 19.7 所示。

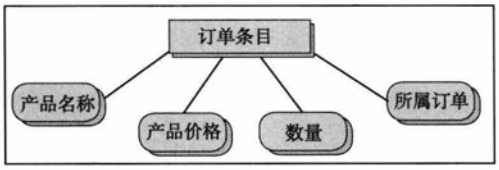


图 19.6 tb_orderitem 表的 E-R 图

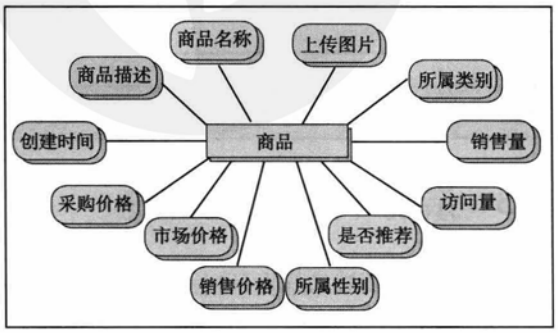


图 19.7 tb_productinfo 表的 E-R 图

5）tb_productcategory（商品类别信息表）的 E-R 图如图 19.8 所示。

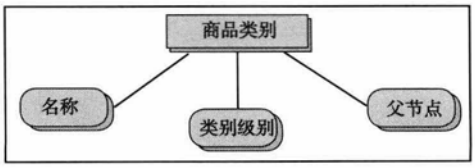


图 19.8 tb_productcategory 表的 E-R 图

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

19.4.2　创建数据库及数据表

创建的数据库名为 db_shop，其中包含 tb_customer、tb_order、tb_orderitem、tb_productinfo、tb_productcategory、tb_user 和 tb_uploadfile 共 7 张数据表。下面将对重要的数据表进行介绍。

（1）tb_customer

该表用于保存会员的注册信息，其结构如表 19.4 所示。

表 19.4　tb_customer 的结构

字 段 名	数 据 类 型	是 否 为 空	是 否 主 键	默 认 值	说　明
id	INT(10)	否	是	NULL	系统自动编号
username	VARCHAR(50)	否	否	NULL	会员名称
password	VARCHAR(50)	否	否	NULL	登录密码
realname	VARCHAR(20)	是	否	NULL	真实姓名
address	VARCHAR(200)	是	否	NULL	地址
email	VARCHAR(50)	是	否	NULL	电子邮件
mobile	VARCHAR(11)	是	否	NULL	电话号码

（2）tb_order

该表用于保存会员的订单信息，其结构如表 19.5 所示。

表 19.5　tb_order 的结构

字 段 名	数 据 类 型	是 否 为 空	是 否 主 键	默 认 值	说　明
id	INT(10)	否	是	NULL	系统自动编号
Name	VARCHAR(50)	否	否	NULL	订单名称
Address	VARCHAR(200)	否	否	NULL	送货地址
mobile	VARCHAR(11)	否	否	NULL	电话
totalPrice	FLOAT	是	否	NULL	采购价格
createTime	DATETIME	是	否	NULL	创建时间
paymentWay	VARCHAR(15)	是	否	NULL	支付方式
orderState	VARCHAR(10)	是	否	NULL	订单状态
customerId	INT(11)	是	否	NULL	会员 ID

（3）tb_orderitem

该表用于保存会员订单的条目信息，其结构如表 19.6 所示。

表 19.6　tb_orderitem 的结构

字 段 名	数 据 类 型	是 否 为 空	是 否 主 键	默 认 值	说　明
id	INT(10)	否	是	NULL	系统自动编号
productId	INT(11)	否	否	NULL	商品 ID
productName	VARCHAR(200)	否	否	NULL	商品名称
productPrice	FLOAT	否	否	NULL	商品价格
amount	INT(11)	是	否	NULL	商品数量
orderId	VARCHAR(30)	是	否	NULL	订单 ID

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（4）tb_productinfo

该表用于保存商品信息，其结构如表 19.7 所示。

表 19.7　tb_productinf 的结构

字 段 名	数 据 类 型	是 否 为 空	是 否 主 键	默 认 值	说　明
id	INT(10)	否	是	NULL	系统自动编号
name	VARCHAR(100)	否	否	NULL	商品名称
description	TEXT	是	否	NULL	商品描述
createTime	DATETIME	是	否	NULL	创建时间
baseprice	FLOAT	是	否	NULL	采购价格
marketprice	FLOAT	是	否	NULL	市场价格
sellprice	FLOAT	是	否	NULL	销售价格
sexrequest	VARCHAR(5)	是	否	NULL	所属性别
commend	BIT(1)	是	否	NULL	是否推荐
clickcount	INT(11)	是	否	NULL	浏览量
sellCount	INT(11)	是	否	NULL	销售量
categoryId	INT(11)	是	否	NULL	商品类别 ID
uploadFile	INT(11)	是	否	NULL	上传文件 ID

（5）tb_user

该表用于保存网站后台管理员信息，其结构如表 19.8 所示。

表 19.8　tb_user 的结构

字 段 名	数 据 类 型	是 否 为 空	是 否 主 键	默 认 值	说　明
id	INT(10)	否	是	NULL	系统自动编号
username	VARCHAR(50)	否	否	NULL	用户名
password	VARCHAR(50)	否	否	NULL	登录密码

19.5　系统文件夹组织结构

为了使项目的程序更加容易管理和维护，在正式编写之前需要设置好项目的系统文件夹的组织结构，将项目中功能类似或者同一个模块的文件放在同一个包中，包名将以模块名称命名。本项目 Java 类的组织结构如图 19.9 所示。



图 19.9　GO 购网络商城 Java 类的文件夹组织结构

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在应用中为了提高系统的安全性，避免用户直接输入地址就可以访问 JSP 页面资源，可以利用项目中的 WEB-INF 文件夹对页面进行保护。众所周知 WEB-INF 文件夹中的文件是不能直接访问的，所以在开发的时候直接将 GO 购网络商城的 JSP 页面放入该文件夹中，这样用户只能通过 Action 才能访问指定的 JSP 页面。视图层 JSP 文件的文件夹组织结构如图 19.10 所示。

说明　WEB-INF 文件夹并不影响页面的转发机制，因为转发是一个内部操作，可以通过 Servlet 或 Action（Action 的本质也是 Servlet）对其进行访问。



图 19.10　GO 购网络商城 JSP 文件夹组织结构

19.6　搭建项目环境

项目开发的第一步是搭建项目环境及项目集成框架等，在此之前需要将 Spring、Struts 2、Hibernate 及系统应用的其他 jar 包导入到项目的 lib 文件下。

19.6.1　配置 Struts 2

在项目的 ClassPath 下创建 struts.xml 文件，其配置代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
   "http://struts.apache.org/dtds/struts-2.3.dtd">
<struts>
   <!-- 前后台公共视图的映射 -->
   <include file="com/lyq/action/struts-default.xml" />
   <!-- 后台管理的 Struts 2 配置文件 -->
   <include file="com/lyq/action/struts-admin.xml" />
   <!-- 前台管理的 Struts 2 配置文件 -->
   <include file="com/lyq/action/struts-front.xml" />
</struts>
```

将 Struts 2 配置文件分为 3 个部分，struts-default.xml 文件为前台和后台公共的视图映射配置文件，其代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
   "http://struts.apache.org/dtds/struts-2.3.dtd">
<struts>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    <!-- OGNL 可以使用静态方法 -->
    <constant name="struts.ognl.allowStaticMethodAccess" value="true"/>
    <package name="shop-default" abstract="true" extends="struts-default">
          <global-results>
                              …<!-- 省略的配置信息 -->
        </global-results>
          <global-exception-mappings>
                <exception-mapping result="error" exception="com.lyq.util.
                AppException"></exception-mapping>
          </global-exception-mappings>
    </package>
</struts>
```

后台管理的 Struts 2 配置文件 struts-admin.xml 主要负责后台用户请求的 Action 和视图映射，其代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
   "http://struts.apache.org/dtds/struts-2.3.dtd">
<struts>
   <!-- 后台管理 -->
   <package name="shop.admin" namespace="/admin" extends="shop-default">
            <!-- 配置拦截器 -->
            <interceptors>
                  <!-- 验证用户登录的拦截器 -->
                  <interceptor name="loginInterceptor"
                        class="com.lyq.action.interceptor.
                        UserLoginInterceptor"/>
                  <interceptor-stack name="adminDefaultStack">
                        <interceptor-ref name="loginInterceptor"/>
                        <interceptor-ref name="defaultStack"/>
                  </interceptor-stack>
          </interceptors>
          <action name="admin_*" class="indexAction" method="{1}">
                  <result name="top">/WEB-INF/pages/admin/top.jsp</result>
                        …<!-- 省略的 Action 配置 -->
                        <interceptor-ref name="adminDefaultStack"/>
          </action>
   </package>
   <package name="shop.admin.user" namespace="/admin/user" extends="shop- default">
          <action name="user_*" method="{1}" class="userAction"></action>
   </package>
     <!-- 栏目管理 -->
     <package name="shop.admin.category" namespace="/admin/product" extends=
     "shop.admin">
          <action name="category_*" method="{1}" class="productCategoryAction">
                …<!-- 省略的 Action 配置 -->
                <interceptor-ref name="adminDefaultStack"/>
          </action>
    </package>
    <!-- 商品管理 -->
    <package name="shop.admin.product" namespace="/admin/product" extends= "shop.
    admin">
          <action name="product_*" method="{1}" class="productAction">
                …<!-- 省略的 Action 配置 -->
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                <interceptor-ref name="adminDefaultStack"/>
            </action>
    </package>
    <!-- 订单管理 -->
    <package name="shop.admin.order" namespace="/admin/product" extends= "shop.
    admin">
            <action name="order_*" method="{1}" class="orderAction">
                ···<!-- 省略的 Action 配置 -->
                <interceptor-ref name="adminDefaultStack"/>
            </action>
    </package>
</struts>
```

前台管理的 Struts 2 配置文件 struts-front.xml 主要负责前台用户请求的 Action 和视图映射，其代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
   "http://struts.apache.org/dtds/struts-2.3.dtd">
<struts>
   <!-- 程序前台 -->
   <package name="shop.front" extends="shop-default">
            <!-- 配置拦截器 -->
            <interceptors>
                <!-- 验证用户登录的拦截器 -->
                <interceptor name="loginInterceptor"
                        class="com.lyq.action.interceptor.
                        CustomerLoginInteceptor"/>
                <interceptor-stack name="customerDefaultStack">
                        <interceptor-ref name="loginInterceptor"/>
                                <interceptor-ref name="defaultStack"/>
                </interceptor-stack>
            </interceptors>
            <action name="index" class="indexAction">
             <result>/WEB-INF/pages/index.jsp</result>
        </action>
   </package>
   <!-- 消费者 Action -->
   <package name="shop.customer" extends="shop-default" namespace= "/customer">
          <action name="customer_*" method="{1}" class="customerAction"></action>
   </package>
   <!-- 商品 Action -->
   <package name="shop.product" extends="shop-default" namespace="/product">
            <action name="product_*" class="productAction" method="{1}">
                ···<!-- 省略的 Action 配置 -->
            </action>
   </package>
   <!-- 购物车 Action -->
   <package name="shop.cart" extends="shop.front" namespace="/product">
            <action name="cart_*" class="cartAction" method="{1}">
                ···<!-- 省略的 Action 配置 -->
                <interceptor-ref name="customerDefaultStack"/>
            </action>
   </package>
   <!-- 订单 Action -->
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    <package name="shop.order" extends="shop.front" namespace="/product">
            <action name="order_*" class="orderAction" method="{1}">
                …<!-- 省略的 Action 配置 -->
                <interceptor-ref name="customerDefaultStack"/>
            </action>
    </package>
</struts>
```

19.6.2 配置 Hibernate

在 Hibernate 的配置文件中配置数据库的连接信息、数据库方言及打印 SQL 语句等属性，其关键代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- 数据库方言 -->
        <property name="hibernate.dialect">org.hibernate.dialect.
        MySQLDialect</property>
        <!-- 数据库驱动 -->
        <property name="hibernate.connection.driver_class">com.mysql.jdbc.
        Driver</property>
        <!-- 数据库连接信息 -->
        <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/
        db_shop</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">111</property>
        <!-- 打印 SQL 语句 -->
        <property name="hibernate.show_sql">true</property>
        <!-- 不格式化 SQL 语句 -->
        <property name="hibernate.format_sql">false</property>
        <!-- 为 Session 指定一个自定义策略 -->
        <property name="hibernate.current_session_context_class">org.
        springframework.orm.hibernate4.SpringSessionContext</property>
        <!-- C3P0 JDBC 连接池 -->
        <property name="hibernate.c3p0.max_size">20</property>
        <property name="hibernate.c3p0.min_size">5</property>
        <property name="hibernate.c3p0.timeout">120</property>
        <property name="hibernate.c3p0.max_statements">100</property>
        <property name="hibernate.c3p0.idle_test_period">120</property>
        <property name="hibernate.c3p0.acquire_increment">2</property>
        <property name="hibernate.c3p0.validate">true</property>
        <!-- 映射文件 -->
        <mapping resource="com/lyq/model/user/User.hbm.xml"/>
        …<!-- 省略的映射文件 -->
    </session-factory>
</hibernate-configuration>
```

说明 C3P0 是一个随 Hibernate 一起分发的开发的 JDBC 连接池，位于 Hibernate 源文件的 lib 目录下。如果在配置文件中设置了 hibernate.c3p0.* 的相关属性，Hibernate 会使用 C3P0ConnectionProvider 来缓存 JDBC 连接。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

19.6.3　配置 Spring

利用 Spring 加载 Hibernate 的配置文件及 Session 管理类，在配置 Spring 时只需要配置 Spring 的核心配置文件 applicationContext-common.xml 即可，其代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:context="http://www.springframework.org/schema/context"
   xmlns:aop="http://www.springframework.org/schema/aop"
   xmlns:tx="http://www.springframework.org/schema/tx"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
         http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
         http://www.springframework.org/schema/context
         http://www.springframework.org/schema/context/spring-context-3.0.xsd
         http://www.springframework.org/schema/aop
         http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
         http://www.springframework.org/schema/tx
         http://www.springframework.org/schema/tx/spring-tx-3.0.xsd">
   <context:annotation-config/>
   <context:component-scan base-package="com.lyq"/>
   <!-- 配置 sessionFactory -->
   <bean id="sessionFactory"
          class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
          <property name="configLocation">
                 <value>classpath:hibernate.cfg.xml</value>
          </property>
   </bean>
   <!-- 配置事务管理器 -->
   <bean id="transactionManager"
          class="org.springframework.orm.hibernate4.HibernateTransactionManager">
          <property name="sessionFactory">
                 <ref bean="sessionFactory" />
          </property>
   </bean>
   <tx:annotation-driven transaction-manager="transactionManager" />
</beans>
```

19.6.4　配置 web.xml

web.xml 的配置文件是项目的基本配置文件，通过该文件设置实例化 Spring 容器、过滤器、Struts 2，以及默认执行的操作，其关键代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/
   ns/javaee/web-app_2_5.xsd"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/
   ns/javaee/web-app_3_0.xsd"
   id="WebApp_ID" version="3.0">
   <display-name>Shop</display-name>
   <!-- 对 Spring 容器进行实例化 -->
   <listener>
          <listener-class>org.springframework.web.context.
          ContextLoaderListener</listener-class>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    </listener>
    <context-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:applicationContext-*.xml</param-value>
    </context-param>
    <!-- OpenSessionInViewFilter 过滤器 -->
    <filter>
            <filter-name>openSessionInViewFilter</filter-name>
            <filter-class>org.springframework.orm.hibernate4.support.
            OpenSessionInViewFilter</filter-class>
    </filter>
    <filter-mapping>
            <filter-name>openSessionInViewFilter</filter-name>
            <url-pattern>/*</url-pattern>
    </filter-mapping>
    <!--Struts 2 配置 -->
    <filter>
            <filter-name>struts2</filter-name>
            <filter-class>org.apache.struts2.dispatcher.ng.filter.
            StrutsPrepareAndExecuteFilter</filter-class>
    </filter>
    <filter-mapping>
            <filter-name>struts2</filter-name>
            <url-pattern>/*</url-pattern>
    </filter-mapping>
    <!-- 设置程序的默认欢迎页面 -->
    <welcome-file-list>
            <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

19.7 公共类设计

在项目中经常会有一些公共类，例如 Hibernate 的初始化类，一些自定义的字符串处理方法等，抽取系统中的公共模块更加有利于代码重用，同时也能提高程序的开发效率，在进行正式开发时首先要进行公共类的编写。下面介绍 GO 购网络商城的公共类。

19.7.1 Hibernate 的 Session 初始化类

Session 对象是 Hibernate 中数据库持久化操作的核心，当多个用户共享一个 Session 时可能会引发数据冲突和混乱，Hibernate 的 Session 管理类要确保 Session 的线程安全。并且创建 Session 的 SessionFactory 是重量级对象，其创建需要耗费大量系统资源。所以为了提高系统的执行效率，将 SessionFactory 的创建放在静态块中，这样在程序运行时就只创建一次。

创建类 HibernateUtils，其关键代码如下：

```
public class HibernateUtils {
    private static SessionFactory factory = null;          // 声明 SessionFactory 对象
    private static final ThreadLocal<Session> threadLocal = new
    ThreadLocal<Session>();                                // 实例化 ThreadLocal 对象
    private static Configuration cfg = new Configuration();
                                                           // 实例化 Configuration 对象
    // 静态块
    static {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            try {
                    cfg.configure();                          // 加载 Hibernate 配置文件
                    factory = cfg.buildSessionFactory();      // 实例化 SessionFactory
            } catch (HibernateException e) {
                    e.printStackTrace();                      // 打印异常信息
            }
    }
    /**
     * 获取 Session 对象
     */
    public static Session getSession() {
            Session session = (Session) threadLocal.get();
            // 从 threadLocal 中获取 Session
            if (session == null || !session.isOpen()) {
            // 判断 session 是否为空或未处于开启状态
                    if (factory == null) {
                            rebuildSessionFactory();
                    }
                    session = (factory != null) ? factory.openSession() : null;
                                                              // 从 factory 开启一个 Session
                    threadLocal.set(session);                 // 将 session 放入 threadLocal 中
            }
            return session;
    }
    /**
     * 获取 SessionFactory 对象
     */
    public static SessionFactory getSessionFactory() {
            return factory;
    }
    /**
     * 关闭 Session
     */
    public static void closeSession() {
            Session session = (Session) threadLocal.get();
                                                              // 从 threadLocal 中获取 Session
            threadLocal.remove();                             // 移除 threadLocal 中的对象
            if (session != null) {
                    if (session.isOpen()) {
                            session.close();                  // 关闭 Session
                    }
            }
    }
    /**
     * 创建 SessionFactory 对象
     */
    public static void rebuildSessionFactory() {
            try {
                    cfg.configure();                          // 加载 Hibernate 配置文件
                    factory = cfg.buildSessionFactory();
                                                              // 实例化 SessionFactory
            } catch (Exception e) {
                    e.printStackTrace();                      // 打印异常信息
            }
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明 为了保证 Session 的线程安全，引入了 ThreadLocal 对象，以保证每次开启的 Session 对象都是最新的，避免多个线程之间共享数据。

19.7.2 泛型工具类

为了将一些公用的持久化方法提取出来，首先需要实现获取实体对象的类型方法，本系统通过创建一个泛型工具类 GenericsUtils 来达到此目的，其代码如下：

```
public class GenericsUtils {
    /**
     * 获取泛型的类型
     * @param clazz
     * @return Class
     */
    @SuppressWarnings("unchecked")
    public static Class getGenericType(Class clazz){
            Type genType = clazz.getGenericSuperclass();          //得到泛型父类
            Type[] types = ((ParameterizedType) genType).getActualTypeArguments();
            if (!(types[0] instanceof Class)) {
              return Object.class;
          }
            return (Class) types[0];
    }
    /**
     * 获取对象的类名称
     * @param clazz
     * @return 类名称
     */
    @SuppressWarnings("unchecked")
    public static String getGenericName(Class clazz){
            return clazz.getSimpleName();
    }
}
```

19.7.3 数据持久化类

本系统利用 DAO 模式封装数据库的基本操作方法，自定义的数据库操作的公共方法如表 19.9 所示。

表 19.9 自定义的数据操作的公共方法

方 法	说 明	参数说明
save(Object obj)	数据保存方法	obj：实体对象
saveOrUpdate(Object obj)	数据添加或保存方法	obj：实体对象
delete(Serializable ... ids)	数据删除方法	ids：删除指定数据的标识
get(Serializable entityId)	查找单条数据加载方法	entityId：查找指定信息的标识
load(Serializable entityId)	查找单条数据加载方法	entityId：查找指定信息的标识
uniqueResult(String hql, Object[] queryParams)	利用 hql 查找单条数据方法	hql：查询的 hql 语句 queryParams：查询的条件参数

根据自定义的数据库操作的公共方法创建接口 BaseDao<T>，关键代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
public interface BaseDao<T> {
    //基本数据库操作方法
    public void save(Object obj);                                    //保存数据
    public void saveOrUpdate(Object obj);                            //保存或修改数据
    public void update(Object obj);                                  //修改数据
    public void delete(Serializable ... ids);                        //删除数据
    public T get(Serializable entityId);                             //加载实体对象
    public T load(Serializable entityId);                            //加载实体对象
    public Object uniqueResult(String hql, Object[] queryParams);
                                                                     //使用 hql 语句操作
}
```

创建 DaoSupport 类，该类继承 BaseDao<T> 接口，在其中实现接口中的自定义方法，其关键代码如下：

```
public class DaoSupport<T> implements BaseDao<T>{
    // 泛型的类型
    protected Class<T> entityClass = GenericsUtils.getGenericType(this.getClass());
    @Override
    public void delete(Serializable ... ids) {
            for (Serializable id : ids) {
                    T t = (T) getSession().load(this.entityClass, id);
                    getSession().delete(t);
            }
    }
    /**
     * 利用 get() 方法加载对象，获取对象的详细信息
     */
    @Transactional(propagation=Propagation.NOT_SUPPORTED,readOnly=true)
    public T get(Serializable entityId) {
            return (T) getSession().get(this.entityClass, entityId);
    }
    /**
     * 利用 load() 方法加载对象，获取对象的详细信息
     */
    @Transactional(propagation=Propagation.NOT_SUPPORTED,readOnly=true)
    public T load(Serializable entityId) {
            return (T) getSession().load(this.entityClass, entityId);
    }
    /**
     * 利用 hql 语句查找单条信息
     */
    @Override
    @Transactional(propagation=Propagation.NOT_SUPPORTED,readOnly=true)
    public Object uniqueResult(final String hql,final Object[] queryParams) {
            Query query=getSession().createQuery(hql);
            setQueryParams(query, queryParams);// 设置查询参数
            return query.uniqueResult();
    }
    /**
     * 获取指定对象的信息条数
     */
    @Transactional(propagation=Propagation.NOT_SUPPORTED,readOnly=true)
    public long getCount() {
            String hql = "select count(*) from " + GenericsUtils.
            getGenericName(this.entityClass);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        return (Long)uniqueResult(hql,null);
    }
    /**
     * 利用 save() 方法保存对象的详细信息
     */
    @Override
    public void save(Object obj) {
        getSession().save(obj);
    }
    @Override
    public void saveOrUpdate(Object obj) {
        getSession().saveOrUpdate(obj);
    }
    /**
     * 利用 update() 方法修改对象的详细信息
     */
    @Override
    public void update(Object obj) {
        getSession().update(obj);
    }
    /**
     * 获取 Session 对象
     * @return
     */
    @Autowired
    @Qualifier("sessionFactory")
    private SessionFactory sessionFactory;
    protected Session getSession(){
        return sessionFactory.getCurrentSession();
    }
}
```

19.7.4 分页设计

本系统应用 Hibernate 的 find 方法实现数据分页，将该方法封装在创建类 DaoSupport 中。

1. 分页实体对象

定义分页的实体对象，并封装分页的基本属性信息和分页过程中使用的获取页码的方法。

```
public class PageModel<T> {
    private int totalRecords;                    //总记录数
    private List<T> list;                        //结果集
    private int pageNo;                          //当前页
    private int pageSize;                        //每页显示多少条
    /**
     * 取得第一页
     */
    public int getTopPageNo() {
    return 1;
    }
    /**
     * 取得上一页
     */
    public int getPreviousPageNo() {
    if (pageNo <= 1) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        return 1;
        }
        return pageNo -1;
        }
        /**
         * 取得下一页
         */
        public int getNextPageNo() {
        if (pageNo >= getTotalPages()) {                    // 如果当前页大于页码
        return getTotalPages() == 0 ? 1 : getTotalPages();  // 返回最后一页
        }
        return pageNo + 1;
        }
        /**
         * 取得最后一页
         */
        public int getBottomPageNo() {
        return getTotalPages() == 0 ? 1 : getTotalPages();
        // 如果总页数为 0 返回 1，反之返回总页数
        }
        /**
         * 取得总页数
         */
        public int getTotalPages() {
        return (totalRecords + pageSize - 1) / pageSize;
        }
        ...                                                 // 省略的 Setter 和 Getter 方法
}
```

在取得上一页页码的 getPreviousPageNo() 方法中，如果当前页的页码数为首页，那么上一页返回的页码数为 1。

在获取最后一页的 getBottomPageNo() 方法中，通过三目运算符判断返回的页码。如果总页数为 0，则返回 1，否则返回总页数。当数据库中没有任何数据时，总页数为 0。

在取得总页码数的 getTotalPages() 方法中，总页的计算公式为（总记录数 + 页面显示记录数 -1）/ 页面显示记录数；另一种方式是使用（总记录数 / 页面显示记录数）计算总页码，如图 19.11 和图 19.12 所示。

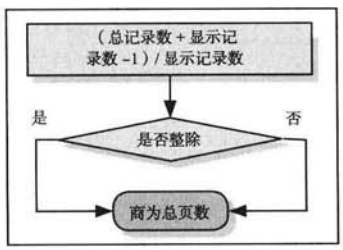


图 19.11 系统中计算总页码的方式

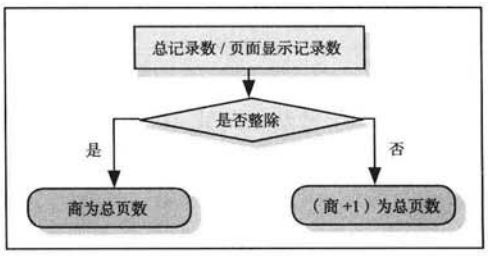


图 19.12 （总记录数 / 页面显示记录数）方式

2. 自定义分页方法

在公共接口中定义的分页方法如表 19.10 所示，这些方法使用相同的分页方法，只是参数不同。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 19.10　自定义分页方法

方　法	说　明	参数说明
getCount()	获取总记录数	无
find(int pageNo, int maxResult)	普通分页方法	pageNo：当前页数
		maxResult：每页显示的记录数
find(int pageNo, int maxResult,String where, Object[] queryParams)	搜索分页方法	pageNo 与 maxResult：同上
		where：查询条件
		queryParams：hql 参数值
find(int pageNo, int maxResult,Map<String, String> orderby)	排序分页方法	其他参数：同上
		orderby：排序的条件参数
find(String where, Object[] queryParams,Map<String, String> orderby, int pageNo, int maxResult)	按条件分页并排序	所有参数：同上

```
public interface BaseDao<T> {
    ...                                                    //基本数据库操作方法
    //分页操作
    public long getCount();                                //获取总信息数
    public PageModel<T> find(int pageNo, int maxResult);   //普通分页操作
    //搜索信息分页方法
    public PageModel<T> find(int pageNo, int maxResult,String where, Object[]
    queryParams);
    //按指定条件排序分页方法
    public PageModel<T> find(int pageNo, int maxResult,Map<String, String> orderby);
    //按指定条件分页和排序的分页方法
    public PageModel<T> find(String where, Object[] queryParams,
                        Map<String, String> orderby, int pageNo, int maxResult);
}
```

在类 DaoSupport 中，实现结构自定义的 find() 分页方法，其简单流程如图 19.13 所示。

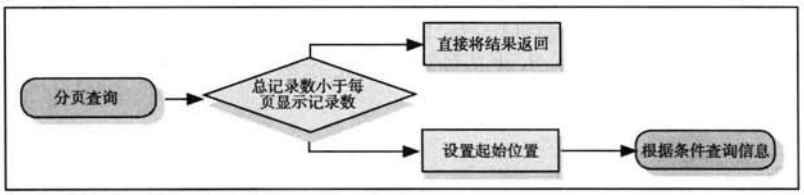


图 19.13　分页方法的简单流程

该方法有 5 个参数，与表 19.10 所示的 5 个方法的参数相同，其代码如下：

```
public PageModel<T> find(final String where, final Object[] queryParams,
                  final Map<String, String> orderby, final int pageNo,
                  final int maxResult) {
    final PageModel<T> pageModel = new PageModel<T>();     //实例化分页对象
    pageModel.setPageNo(pageNo);                           //设置当前页数
    pageModel.setPageSize(maxResult);                      //设置每页显示记录数
    getTemplate().execute(new HibernateCallback() {        //执行内部方法
    @Override
    public Object doInHibernate(Session session) throws HibernateException,
    SQLException {
    String hql = new StringBuffer().append("from ")        //添加 form 字段
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                .append(GenericsUtils.getGenericName(entityClass))
                                                        //添加对象类型
                .append(" ")                            //添加空格
                .append(where == null ? "" : where)
                //如果where为null就添加空格，反之添加where
                .append(createOrderBy(orderby))     //添加排序条件参数
                .toString();                        //转换为字符串
    Query query = session.createQuery(hql);         //执行查询
    setQueryParams(query,queryParams);              //为参数赋值
    List<T> list = null;                            //定义List对象
    // 如果maxResult<0，则查询所有
    if(maxResult < 0 && pageNo < 0){
    list = query.list();                                    //将查询结果转换为List对象
    }else{
    list = query.setFirstResult(getFirstResult(pageNo, maxResult))
                                                        //设置分页起始位置
                    .setMaxResults(maxResult)           //设置每页显示的记录数
                .list();                                //将查询结果转换为List对象
                //定义查询总记录数的hql语句
                hql = new StringBuffer().append("select count(*) from ")
                                                        //添加hql语句
                .append(GenericsUtils.getGenericName(entityClass))
                                                        //添加对象类型
                .append(" ")                            //添加空格
                .append(where == null ? "" : where)
                //如果where为null就添加空格，反之添加where
                .toString();                            //转换为字符串
                query = session.createQuery(hql);       //执行查询
                setQueryParams(query,queryParams);      //设置hql参数
                int totalRecords = ((Long) query.uniqueResult()).intValue();
                                                        //类型转换
                pageModel.setTotalRecords(totalRecords);  //设置总记录数
    }
    pageModel.setList(list);                    //将查询的list对象放入实体对象中
                return null;
    }
    });
    return pageModel;                                   //返回分页的实体对象
}
```

在上述代码中使用了StringBuffer()的append()方法拼接查询的hql语句，通过toString()方法将拼接的hql语句转换为字符串。通过getFirstResult()方法获取分页的起始位置，其代码如下：

```
protected int getFirstResult(int pageNo,int maxResult){
    int firstResult = (pageNo-1) * maxResult;
    return firstResult < 0 ? 0 : firstResult;
}
```

代码中的分页起始位置为（当前页码 -1）× 页面显示记录数，如果页面起始位置小于0，则返回0，否则返回程序计算的起始位置。

19.8　登录注册模块设计

如果要提高网站的安全性，防止非法用户进入网站，可以让用户进入网站前先进行注册，

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

注册成功的用户才可以通过购物车购买商品。用户注册在大多数网站中都是不可缺少的功能，也是用户参与网站活动最为直接的桥梁。通过用户注册，可以有效地对用户信息进行采集，并将合法的用户信息保存到指定的数据表中，当用户注册操作完毕后，通常情况下，该注册用户将直接登录该网站。

由于 GO 购网络商城主要分为前台和后台两个部分，所以登录也为分为前台登录和后台登录两个部分的功能，前台的登录针对的是在 GO 购网络商城注册的会员，后台操作主要针对的是网站的管理员，而注册模块主要针对的就是前台想进行购物的游客。登录注册模块的框架如图 19.14 所示。

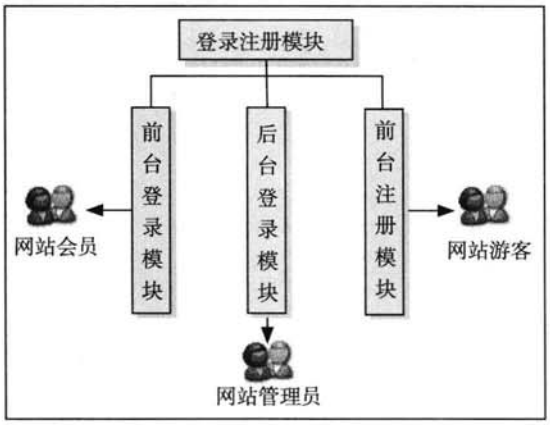


图 19.14　登录注册模块的框架



图 19.15　会员注册页面

19.8.1　实现注册模块

在安全注册与登录操作过程中严格验证表单内容，以提高网站的安全性，防止非法用户进入网站。

本模块中的注册页面为 customer_reg.jsp，如图 19.15 所示。其中，用户名要求 5 ~ 32 个字符，密码表单与重复输入密码表单必须一致；邮箱地址表单必须是正确的地址，通过 Struts 2 的校验器校验；住址与手机两个表单在用户购买商品生成订单时直接获取相关的送货信息。

实现注册表单页面的关键代码如下：

```
<s:form action="customer_save" name="/user" method="post">
<s:fielderror></s:fielderror>
用 户 名：<s:textfield name="username" cssClass="bian"></s:textfield>*
密　码：<s:password name="password" cssClass="bian"></s:password>*
确认密码：<s:password name="repassword" cssClass="bian"></s:password>*
邮箱地址：<s:textfield name="email" cssClass="bian"></s:textfield>*
住　址：<s:textfield name="address" cssClass="bian"></s:textfield>
手　机：<s:textfield name="mobile" cssClass="bian"></s:textfield>
   <s:submit value="登　录" type="image" src="%{context_path}/css/images/dl_08.
   gif"></s:submit>
   </s:form>
```

1．表单验证

在本模块中使用 XML 文件对表单中的信息进行合法性验证，利用 requiredsting 校验器对

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

CustomerAction 类中的字段进行非空验证，利用 stringlength 校验器对 CustomerAction 类中的字段长度进行验证，利用 email 校验器对邮箱地址的格式进行验证。在 CustomerAction 类的包下新建 XML 文件，关键代码如下：

```
CustomerAction-customer_save-validation.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE validators PUBLIC  "-//OpenSymphony Group//XWork Validator 1.0.3//EN"
   "http://www.opensymphony.com/xwork/xwork-validator-1.0.3.dtd" >
<validators>
   <field name="username">
            <field-validator type="requiredstring" >
                  <message>用户名不能为空 </message>
            </field-validator>
            <field-validator type="stringlength">
                  <param name="minLength">5</param>
                  <param name="maxLength">32</param>
                  <message>用户名长度必须在 ${minLength} 到 ${maxLength} 之间 </
                  message>
            </field-validator>
   </field>
   <field name="password">
            <field-validator type="requiredstring">
                  <message>密码不能为空 </message>
            </field-validator>
            <field-validator type="stringlength">
                  <param name="minLength">6</param>
                  <message>密码长度必须在 ${minLength} 位以上 </message>
            </field-validator>
   </field>
   <field name="repassword">
            <field-validator type="requiredstring" short-circuit="true">
                  <message>确认密码不能为空 </message>
            </field-validator>
            <field-validator type="fieldexpression">
                  <param name="expression">password == repassword</param>
                  <message>两次密码不一致 </message>
            </field-validator>
   </field>
   <field name="email">
            <field-validator type="requiredstring">
                  <message>邮箱不能为空 </message>
            </field-validator>
            <field-validator type="email">
                  <message>邮箱格式不正确 </message>
            </field-validator>
   </field>
</validators>
```

注意

Struts 2 的另一种表单验证方式是 Aciton 类重写 ActionSupport 类的 validate() 方法，应用该方法的步骤如下。

1）通过转换器将请求参数转换为相应的 bean 属性。

2）判断转换过程是否出现异常。如果有，则将其保存在 ActionContext 中，conversionError 拦截器再封装为 fieldError，否则执行下一步操作。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

注意

3）利用反射（Reflection）来调用 validateXxx() 方法（其中 Xxx 表示 Action 的方法名）。
4）调用 validate() 方法。
5）如果经过上述步骤没有出现 fieldError，则调用 Action 方法；否则会跳过 Action 方法，通过国际化（指错误信息）将 fieldError 输出到页面。

2. 保存注册信息

当用户单击会员注册页面中的“注册”超链接时，系统将会发送一个 customer_reg.html 的 URL 请求。该请求将会执行 CustomerAction 类中的 save() 方法，其首先判断用户名是否可用，如果可用，则将注册信息保存在数据库中，否则返回错误信息，其关键代码如下：

```
public String save() throws Exception{
   boolean unique = customerDao.isUnique(customer.getUsername());
                                                          //判断用户名是否可用
   if(unique){                                            //如果用户名可用
   customerDao.save(customer);                            //保存注册信息
   return CUSTOMER_LOGIN;                                 //返回会员登录页面
   }else{
   throw new AppException("此用户名不可用");               //否则返回页面错误信息
   }
}
```

19.8.2 实现登录模块

前台与后台的登录验证方法基本一致，只是前台登录保存的是登录会员信息，后台登录保存的是网站管理员的基本信息。前台会员登录页面如图 19.16 所示，后台管理员登录页面如图 19.17 所示。



图 19.16 前台会员登录页面



图 19.17 后台管理员登录页面

前台与后台的登录页面代码的实现方式相同，这里以前台登录页面为例，其关键代码如下：

```
<s:fielderror></s:fielderror>
<s:form action="customer_logon" namespace="/customer" method="post">
会员名: <s:textfield name="username" cssClass="bian" size="18"> </s:text field>
密  码: <s:password name="password" cssClass="bian" size="18"> </s:pass word>
<s:submit value="登  录" type="image" src="%{context_path}/css/images/dl_06.
gif"></s:submit>
<s:a action="customer_reg" namespace="/customer">
<img src="${context_path}/css/images/dl_08.gif" width="68" height ="24" /></s:a>
</s:form>
```

在登录验证的过程中通过页面中获取的用户名和密码作为查询条件在用户信息表中查找

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

条件匹配的用户信息，如果返回的结果集不为空，说明验证通过，反之失败。前台登录验证方法关键代码如下：

```
public String logon() throws Exception{
   //验证用户名和密码是否正确
   Customer loginCustomer = customerDao.login(customer.getUsername(), customer.
getPassword());
   if(loginCustomer != null){                          //如果通过验证
session.put("customer", loginCustomer);                //将登录会员信息保存在 Session 中
   }else{                                              //验证失败
   addFieldError("", "用户名或密码不正确！");           //返回错误信息
   return CUSTOMER_LOGIN;                              //返回会员登录页面
   }
   return INDEX;                                       //返回网站首页
}
```

后台登录验证方法关键代码如下：

```
public String logon() throws Exception{
   //验证用户名和密码
   User loginUser = userDao.login(user.getUsername(), user.getPassword());
   if(loginUser != null){                              //通过验证
   session.put("admin", loginUser);                    //将管理员信息保存在 Session 对象中
   }else{
   addFieldError("", "用户名或密码不正确！");           //返回错误提示信息
   return USER_LOGIN;                                  //返回后台登录页面
   }
   return MANAGER;                                     //返回后台管理页面
}
```

前后台公共的 login() 方法以用户名和密码作为查询条件并返回查询的用户对象，其关键代码如下：

```
public User login(String username, String password) {
   if(username != null && password != null){           //如果用户名和密码不为空
   String where = "where username=? and password=?";   //设置查询条件
   Object[] queryParams = {username,password};         //设置参数对象数组
   List<User> list = find(-1, -1, where, queryParams).getList();
                                                       //执行查询方法
   if(list != null && list.size() > 0){                //如果 list 集合不为空
   return list.get(0);                                 //返回 List 中的第 1 个存储对象
   }
   }
   return null;                                        //返回空值
}
```

在数据库设计中已经保证了用户名的唯一性（在数据表中将用户名作为表的主键），所以查询结果只能返回一个 Object 对象。但是为了考虑程序的健壮性，返回 List 集合更有利于程序的扩展，降低出错的概率。

19.9　前台商品信息查询模块设计

前台商品信息查询模块划分为 5 个模块，主要包括商品分类查询、人气商品查询、热销商品查询、推荐商品查询以及商品的模糊查询，如图 19.18 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

19.9.1 实现商品分类查询

在前台的首页商品展示中首先展现的是商品类别的分类显示，方便用户按类别查询商品。商品类别分类显示的效果如图 19.19 所示。

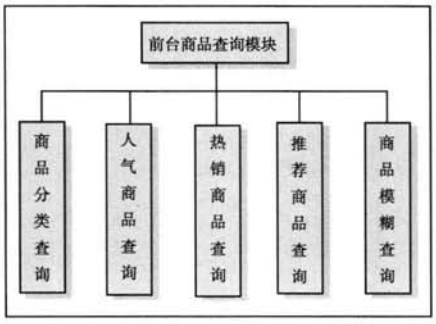


图 19.18 前台商品信息查询模块的框架



图 19.19 商品类别分类显示的效果

在程序中可以通过迭代方式按所属级别分层显示所有的商品类别，第一次查询所有无父节点的类别信息，第二次遍历其子节点，最后遍历叶子节点，其流程如图 19.20 所示。

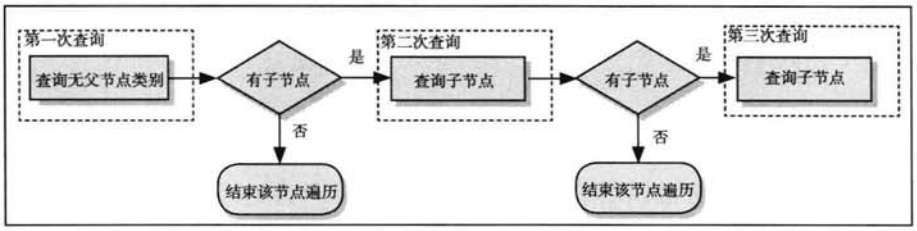


图 19.20 商品类别分类显示的流程

（1）查询一级节点

通过公共模块持久化类中封装的 find() 方法查询所有的一级节点，在首页的 Action 请求 IndexAction 的 execute() 方法中调用封装的 find() 方法，其关键代码如下：

```
public String execute() throws Exception {
    // 查询所有类别
    String where = "where parent is null";
    categories = categoryDao.find(-1, -1, where, null).getList();
    …                                                    // 省略的 Setter 和 Getter 方法
}
```

find() 方法有 4 个参数，其中“-1”参数为当前页数和每页显示的记录数；“where”参数为查询条件；“null”参数为数据排序的条件。find() 方法会根据提供的两个“-1”参数执行以下代码：

```
// 如果 maxResult<0，则查询所有
if(maxResult < 0 && pageNo < 0){
    list = query.list();                          // 将查询结果转换为 List 对象
}
```

（2）页面遍历

通过 Struts 2 的 <s:iterator> 标签遍历查询的结果集 categories，然后将超链接的 Action 请

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

求和参数赋值给三级节点，其关键代码如下：

```
<!-- 类别 -->
<s:iterator value="categories">
…<!-- 省略的布局及样式代码 -->
   <!-- 二级 -->
   <s:if test="!children.isEmpty">
   <s:iterator value="children">
   <!-- 三级 -->
   <s:if test="!children.isEmpty">
   <span>
   <s:iterator value="children">
   <s:a action="product_getByCategoryId" namespace="/product">
   <s:param name="category.id" value="id"></s:param>
   <s:property value="name" escape="false"/>
   </s:a>
   </s:iterator>
   </span>
   </s:if>
   </s:iterator>
   </s:if>
   …<!-- 省略的布局及样式代码 -->
</s:iterator>
```

19.9.2　实现商品搜索

当搜索表单中没有输入数据时，单击“搜索”按钮，查询数据表中的所有数据；当在关键字文本框中输入要搜索的内容后，单击“搜索”按钮可以按关键字查询数据表中的所有数据，如图 19.21 所示。



图 19.21　搜索商品

搜索商品的方法封装在 ProductAction 类中，通过 hql 的 like 条件语句实现商品的模糊查询功能，其关键代码如下：

```
public String findByName() throws Exception {
   if(product.getName() != null){
      String where = "where name like ?";                          //查询的条件语句
      Object[] queryParams = {"%" + product.getName() + "%"};  //为参数赋值
   pageModel = productDao.find(pageNo, pageSize, where, queryParams );
                                                                   //执行查询方法
   }
   return LIST;                                                    //返回列表首页
}
```

注意　程序中返回的“LIST”并不是真正的 List 集合，而是前台显示商品列表信息页面的视图名。

在商品的列表页中通过 Struts 2 的 <s:iterator> 标签遍历返回的商品 List 集合，其关键代码

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

如下：

```
<s:iterator value="pageModel.list">
          <table border="0" width="100%" cellpadding="0" cellspacing="0">
                    <tr><td rowspan="5" width="160">
                    <s:a action="product_select" namespace="/product">
                    <s:param name="id" value="id"></s:param>
                    <img width="150" height="150"src="<s:property
                    value="#request.get( 'javax.servlet.forward.context_ path')
                    "/>/upload
                    /<s:property value="uploadFile.path"/>">
                 </s:a></td>
          </tr><tr bgcolor="#f2eec9">
                    <td align="right" width="90">商品名称：</td>
                    <td><s:a action="product_select" namespace="/product">
                              <s:param name="id" value="id"></s:param>
                              <s:property value="name" />
                    </s:a></td>
          </tr> <tr>
                              <td align="right" width="90">市场价格：</td>
                              <td><font style="text-decoration: line-through;">
                              <s:property value="marketprice" /> </font></td>
          </tr><tr bgcolor="#f2eec9">
                    <td align="right" width="90">GO购网络价格：</td>
                    <td><s:property value="sellprice" />
                              <s:if test="sellprice <= marketprice">
                              <font color="red">节省
                              <s:property value="marketprice-sellprice" /></font>
                              </s:if></td>
                    </tr><tr>
                              <td colspan="2" align="right">
                              <s:a action="product_select" namespace="/product">
                              <s:param name="id" value="id"></s:param>
                             <img src="${context_path}/css/images/gm_06.gif"
                             width="136"
                             height="32" />
                            </s:a></td>
                    </tr>
           </table>
</ s:iterator >
```

19.9.3 实现前台查询其他商品

人气商品推荐模块（如图 19.22 所示）、推荐商品模块（如图 19.23 所示）和热销商品模块（如图 19.24 所示）3 种查询方式的实现基本相同，都是通过条件语句执行排序查询。



图 19.22 人气商品推荐模块



图 19.23 推荐商品模块



图 19.24 热销商品模块

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

1．人气商品模块

人气商品的定义是基于商品浏览量的最大数，本系统显示筛选出商品中浏览量最多的几种商品。商品结果集中的数据按照商品浏览量倒序排列，实现方法封装在 ProductAction 类中，其关键代码如下：

```
public String findByClick() throws Exception{
    Map<String, String> orderby = new HashMap<String, String>();
                                                    //定义 Map 集合
    orderby.put("clickcount", "desc");              //为 Map 集合赋值
    pageModel = productDao.find(1, 8, orderby );    //执行查找方法
    return "clickList";                             //返回 product_click_list.jsp 页面
}
```

在调用的 find() 方法中使用了 3 个参数，即显示的起始位置、显示记录数和商品信息排序的条件。程序最终返回到 product_click_list.jsp 页面，其中的关键代码如下：

```
<s:set var="context_path" value="#request.get('javax.servlet.forward.context_
path')"></s:set >
<table width="193" height="23" border="0" cellpadding="0" cellspacing="0">
    <s:iterator value="pageModel.list">
    <tr>
    <td width="187" valign="middle">
    <img src="${context_path}/css/images/h_32.gif" width="20" height="17" />
    <s:a action="product_select" namespace="/product">
    <s:param name="id" value="id"></s:param>
    <s:property value="name"/>（人气：
    <span class="red"><s:property value="clickcount"/></span>）
    </s:a>
    </td>
    </tr>
</s:iterator>
</table>
```

2．推荐商品和热销商品模块

推荐商品和热销商品模块通过 hql 的排序语句实现，推荐商品为商品推荐字段 commend 为 true 的商品并且按商品销量的倒序排列，实现的关键代码如下：

```
public String findByCommend() throws Exception{
    Map<String, String> orderby = new HashMap<String, String>();
                                                    //定义 Map 集合
    orderby.put("sellCount", "desc");               //为 Map 集合赋值
    String where = "where commend = ?";             //设置条件语句
    Object[] queryParams = {true};                  //设置参数值
    pageModel = productDao.find(where, queryParams, orderby, pageNo, pageSize);
                                                    //执行查询方法
    return "findList";                              //返回推荐商品页面
}
```

热销商品为销售量较多的商品，只需按照商品销量的倒序排列，并以分页方式取出前 6 条信息，实现的关键代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
public String findBySellCount() throws Exception{
   Map<String, String> orderby = new HashMap<String, String>();
                                                    //定义 Map 集合
   orderby.put("sellCount", "desc");                //为 Map 集合赋值
   pageModel = productDao.find(1, 6, orderby );     //执行查询方法
   return "findList";                               //返回热销商品页面
}
```

在 Struts 2 的前台 Action 配置文件 struts-front.xml 中配置前台商品管理模块的 Action 及视图映射关系，关键代码如下：

```
<!-- 商品 Action -->
<package name="shop.product" extends="shop-default" namespace="/product">
   <action name="product_*" class="productAction" method="{1}">
   <result name="list">/WEB-INF/pages/product/product_list.jsp</result>
   <result name="select">/WEB-INF/pages/product/product_select.jsp</result>
   <result name="clickList">/WEB-INF/pages/product/product_click_list.jsp </result>
   <result name="findList">/WEB-INF/pages/product/product_find_list.jsp</result>
   </action>
</package>
```

19.10 购物车模块设计

购物车是商务网站中必不可少的功能，GO 购网络商城购物车实现的主要功能有添加选购的新商品、自动更新选购的商品数量、清空购物车、自动调整商品总价格以及生成订单信息等。本模块实现的购物车的功能流程如图 19.25 所示。

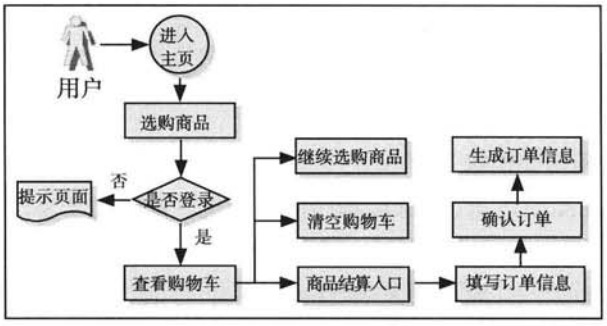


图 19.25 购物车的功能流程

19.10.1 实现购物车的基本功能

购物车的功能基于 Session 变量实现，Session 充当了一个临时信息存储平台。当其失效后，保存的购物车信息也将全部丢失。

（1）在购物车中添加商品

登录会员浏览商品详细信息并单击页面中如图 19.26 所示的“立即购买”超链接后，会将该商品放入购物车内，如图 19.27 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 19.26　“立即购买”超链接



图 19.27　放入购物车内的商品信息

在购物车中添加商品时，首先要获取商品 id。如果购物车中存在相同 id 值，则修改该商品的数量，将数量自动加 1，否则添加新的商品购买信息。添加商品信息的方法封装在 CartAction 类中，其关键代码如下：

```
public String add() throws Exception {
   if(productId != null && productId > 0){
   Set<OrderItem> cart = getCart();                    //获取购物车
   // 标记添加的商品是否是同一件商品
   boolean same = false;                               //定义 same 布尔变量
   for (OrderItem item : cart) {                       //遍历购物车中的信息
   if(item.getProductId() == productId){
            // 购买相同的商品，更新数量
            item.setAmount(item.getAmount() + 1);
            same = true;                               //设置 same 变量为 "true"
            }
   }
   // 不是同一种商品
   if(!same){
   OrderItem item = new OrderItem();                   //实例化订单条目信息实体对象
   ProductInfo pro = productDao.load(productId);//加载商品对象
   item.setProductId(pro.getId());                     //设置 id
   item.setProductName(pro.getName());                 //设置商品名称
   item.setProductPrice(pro.getSellprice());           //设置商品销售价格
   item.setProductMarketprice(pro.getMarketprice());
                                                       //设置商品市场价格
   cart.add(item);                                     //将信息添加到购物车中
   }
   session.put("cart", cart);                          //将购物车保存在 Session 对象中
   }
   return LIST;
}
```

程序运行结束后返回订单条目信息的列表页面，即 cart_list.jsp，其关键代码如下：

```
<s:iterator value="#session.cart">
   <s:set value="%{#sumall +productPrice*amount}" var="sumall" />
   …<!-- 省略的布局代码 -->
   <td width="213" height="30" align="center">
   <s:property value="productName" /></td>
   <td width="130" align="center">
   <span style="text-decoration: line-through;"> ￥
   <s:property value="productMarketprice" /> 元 </span></td>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    <td width="130" align="center">￥
    <s:property value="productPrice" />元<br>为您节省：￥
<s:propertyvalue="productMarketprice*amount - productPrice*amount" />元</td>
    <td width="104" align="center" class="red">
              <s:property value="amount" /></td>
    <td width="111" align="center"><s:a action="cart_delete" namespace= "/product">
              <s:param name="productId" value="productId"></s:param>
              <img src="${context_path}/css/images/zh03_03.gif" width="52" height
              ="23" />
    </s:a></td>
    …<!-- 省略的布局代码 -->
</s:iterator >
```

（2）删除购物车中指定商品订单条目信息

单击购物车中的某个商品的订单条目信息后（如图 19.28 所示的“删除”超链接），系统将自动清除该商品的订单条目信息。



图 19.28 “删除”超链接

单击“删除”链接后，URL 发送一个 cart_delete.html 的请求，该请求执行 CartAction 中的 delete() 方法，其关键代码如下：

```
public String delete() throws Exception {
    Set<OrderItem> cart = getCart();                    // 获取购物车
    // 此处使用 Iterator，否则出现 java.util.ConcurrentModificationException
    Iterator<OrderItem> it = cart.iterator();
    while(it.hasNext()){                                // 使用迭代器遍历商品订单条目信息
    OrderItem item = it.next();
    if(item.getProductId() == productId){
    it.remove();                                        // 移除商品订单条目信息
    }
    }
    session.put("cart", cart);                          // 将清空后的信息重新放入 Session 中
    return LIST;                                        // 返回购物车页面
}
```

（3）清空购物车

单击购物车页面中的“清空”按钮，向服务器发送一个 cart_clear.html 的 URL 请求。该请求执行 CartAction 类中的 clear() 方法，其关键代码如下：

```
public String clear() throws Exception {
    session.remove("cart");                             // 移除信息
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    return LIST;                                          //返回订单列表页面
}
```

（4）查找购物信息

单击首页顶部如图 19.29 所示的“我的购物车”链接，可以查看购物车的相关信息。



图 19.29　“我的购物车”超链接

单击“我的购物车”超链接后发送一个 cart_list.html 的 URL 请求，该请求执行 CartAction 中的 list() 方法，其关键代码如下：

```
public String list() throws Exception {
    return LIST;                                          //返回购物车页面
}
```

在购物车页面中通过 Struts 2 的 <s:iterator> 标签遍历 Session 对象中购物车的相关信息，在程序模块中，只需要返回购物车页面即可。

在 Struts 2 的前台 Action 配置文件 struts-front.xml 中配置购物车管理模块的 Action 及视图映射关系，其关键代码如下：

```
<!-- 购物车 Action -->
<package name="shop.cart" extends="shop.front" namespace="/product">
    <action name="cart_*" class="cartAction" method="{1}">
    <result name="list">/WEB-INF/pages/cart/cart_list.jsp</result>
    <interceptor-ref name="customerDefaultStack"/>
    </action>
</package>
```

19.10.2　实现订单的相关功能

要结算选购的商品，首先要生成一个订单，其中包括收货人信息、送货方式、支付方式、购买的商品及订单总价格。用户在购物车中单击“收银台结账”超链接后打开填写订单页面 order_add.jsp，如图 19.30 所示。

（1）下订单

单击购物车“收银台结账”超链接后发送一个 order_add.html 的 URL 请求，该请求执行 OrderAction 类中的 add() 方法，将用户的基本信息从 Session 对象中取出并添加到订单表单中的指定位置，然后跳转到“我的订单”页面，其关键代码如下：

```
public String add() throws Exception {
    order.setName(getLoginCustomer().getUsername());      //设置收货人姓名
    order.setAddress(getLoginCustomer().getAddress());    //设置收货人地址
    order.setMobile(getLoginCustomer().getMobile());      //设置收货人电话
    return ADD;                                           //返回"我的订单"页面
}
```

（2）订单确认

单击我的订单页面中的“付款”按钮，打开“订单确认”页面，如图 19.31 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 19.30　填写订单页面

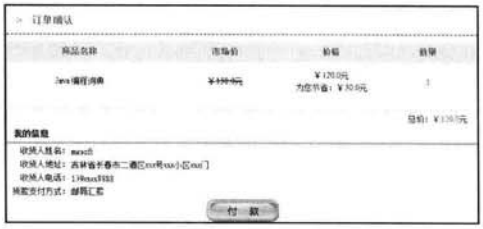


图 19.31　“订单确认”页面

其中显示订单的条目信息，即用户购买商品的信息清单，以便用户确认。

单击“付款”按钮发送一个 order_ confirm.html 的 URL 请求，该请求执行 OrderAction 类中的 confirm() 方法，其关键代码如下：

```
public String confirm() throws Exception {
    return "confirm";                                    //返回"订单确认"页面
}
```

（3）订单保存

单击“订单确认”页面中的“付款”按钮触发 OrderAction 类中的 save() 方法，把订单信息保存到数据库中，其关键代码如下：

```
public String save() throws Exception {
    if(getLoginCustomer() != null){                      //如果用户已登录
    order.setOrderId(StringUitl.createOrderId());        // 设置订单号
    order.setCustomer(getLoginCustomer());               // 设置所属用户
    Set<OrderItem> cart = getCart();                     // 获取购物车
    // 依次更新订单项中的商品销售数量
    for(OrderItem item : cart){                          //遍历购物车中的订单条目信息
    Integer productId = item.getProductId();             //获取商品 id
    ProductInfo product = productDao.load(productId);    //加载商品对象
    product.setSellCount(product.getSellCount() + item.getAmount());
                                                         //更新商品销售数量
    productDao.update(product);                          //修改商品信息
    }
    order.setOrderItems(cart);                           // 设置订单项
        order.setOrderState(OrderState.DELIVERED);       // 设置订单状态
    float totalPrice = 0f;                               // 计算总额的变量
    for (OrderItem orderItem : cart) {                   //遍历购物车中的订单条目信息
    totalPrice += orderItem.getProductPrice() * orderItem.getAmount();
                                                         //商品单价 × 商品数量
    }
    order.setTotalPrice(totalPrice);                     //设置订单的总价格
    orderDao.save(order);                                //保存订单信息
    session.remove("cart");                              //清空购物车
}
    return findByCustomer();                             //返回消费者订单查询的方法
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

执行 save() 方法后返回订单查询的 findByCustomer() 方法，其中以登录用户的 id 为查询条件查询该用户的所有订单信息，其关键代码如下：

```
public String findByCustomer() throws Exception {
   if(getLoginCustomer() != null){                          //如果用户已登录
   String where = "where customer.id = ?";                  //将用户 id 设置为查询条件
      Object[] queryParams = {getLoginCustomer().getId()};
                                                            //创建对象数组
   Map<String, String> orderby = new HashMap<String, String>(1);
                                                            //创建 Map 集合
   orderby.put("createTime", "desc");                       //设置排序条件及方式
   pageModel = orderDao.find(where, queryParams, orderby , pageNo, pageSize);
                                                            //执行查询方法
   }
   return LIST;                                             //返回订单列表页面
}
```

查询后返回订单列表页面 order_list.jsp，如图 19.32 所示。



图 19.32　订单列表页面

在 Struts 2 的前台 Action 配置文件 struts-front.xml 中，配置前台订单管理模块的 Action 及视图映射关系，关键代码如下：

```
<!-- 订单 Action -->
<package name="shop.order" extends="shop.front" namespace="/product">
   <action name="order_*" class="orderAction" method="{1}">
   <result name="add">/WEB-INF/pages/order/order_add.jsp</result>
   <result name="confirm">/WEB-INF/pages/order/order_confirm.jsp</result>
   <result name="list">/WEB-INF/pages/order/order_list.jsp</result>
   <result name="error">/WEB-INF/pages/order/order_error.jsp</result>
   <interceptor-ref name="customerDefaultStack"/>
   </action>
</package>
```

19.11　后台商品管理模块设计

GO 购网络商城网站的商品管理模块主要实现的是商品信息查询、修改商品信息、删除商品信息以及添加商品信息。后台商品管理模块的框架如图 19.33 所示。

19.11.1　实现商品管理功能

在商品管理的基本模块中，包括商品的查询、修改、删除及添加，下面分别介绍。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（1）查询商品信息

在 GO 购网络商城的后台管理页面中，单击左侧导航栏中的“查看所有商品”超链接，显示所有商品的查询页面，如图 19.34 所示。

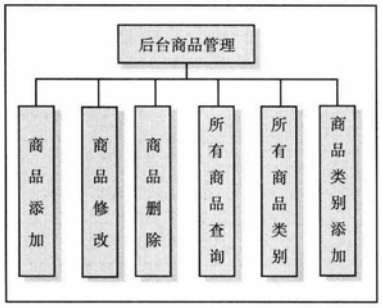


图 19.33　后台商品管理模块的框架

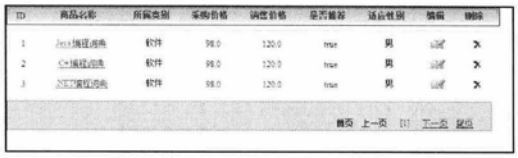


图 19.34　所有商品的查询页面

该页面实现的关键代码如下：

```
< table width="693" height="29" border="0" class="word01"><tr>
   <td width="37" height="27" align="center">ID</td>
   <td width="120" align="center"> 商品名称 </td>
   <td width="78" align="center"> 所属类别 </td>
   <td width="79" align="center"> 采购价格 </td>
   <td width="79" align="center"> 销售价格 </td>
   <td width="79" align="center"> 是否推荐 </td>
   <td width="79" align="center"> 适应性别 </td>
   <td width="52" align="center"> 编辑 </td>
   <td width="52" align="center"> 删除 </td>
   </tr></table>
<div id="right_mid"><div id="tiao"><table width="693" height="29" border="0">
   <s:iterator value="pageModel.list">
   <tr>
   <td width="37" height="27" align="center"><s:property value="id" /></td>
   <td width="120" align="center"><s:a action="product_edit" namespace= "/admin/
   product">
   <s:param name="id" value="id"></s:param><s:property value="name" /></s:a> </td>
   <td width="78" align="center"><s:property value="category.name" /></td>
   <td width="79" align="center"><s:property value="baseprice" /></td>
   <td width="79" align="center"><s:property value="sellprice" /></td>
   <td width="79" align="center"><s:property value="commend" /></td>
   <td width="79" align="center"><s:property value="sexrequest.name" /></td>
   <td width="52" align="center"><s:a action="product_edit" namespace= "/admin/
   product">
   <s:param name="id" value="id"></s:param>
   <img src="${context_path}/css/images/rz_15.gif" width="21" height="16" /> </
   s:a></td>
   <td width="52" align="center"><s:a action="product_del" namespace ="/admin/
   product">
   <s:param name="id" value="id"></s:param>
   <img src="${context_path}/css/images/rz_17.gif" width="15"height="16" /> </
   s:a></td>
   </tr>
   </s:iterator>
</table>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

单击“查看所有商品”超链接，发送一个 product_list.html 的 URL 请求，该请求执行 ProductAction 类中的 list() 方法。该类继承了 BaseAction 类和 ModelDriven 接口，其关键代码如下：

```
public String list() throws Exception{
    pageModel = productDao.find(pageNo, pageSize);        //调用公共的查询方法
   return LIST;                                           //返回后台商品列表页面
}
```

当用户单击列表中的商品名称超链接或列表中的按钮，将进入商品信息的编辑页面，如图 19.35 所示。



图 19.35　商品信息编辑页面

在其中可以修改商品的信息，该操作触发商品详细信息的查找方法，即 ProductAction 类中的 edit() 方法。该方法将以商品的 id 值作为查询条件，其关键代码如下：

```
public String edit() throws Exception{
   this.product = productDao.get(product.getId());       //执行封装的查询方法
   createCategoryTree();                                 //生成商品的类别树
   return EDIT;                                          //返回商品信息编辑页面
}
```

商品信息编辑页面的关键代码如下：

```
商品名称：<s:textfield name="name"></s:textfield>
   <img width="270" height="180" border="1" src="<s:property value="#request.
   get('javax.servlet.forward.context_path')"/>
   /upload/<s:property value="uploadFile.path"/>">
   选择类别：<s:select name="category.id" list="map" value="category.id"> </s:select>
采购价格：<s:textfield name="baseprice"></s:textfield>
市场价格：<s:textfield name="marketprice"></s:textfield>
销售价格：<s:textfield name="sellprice"></s:textfield>
是否为推荐：<s:radio name="commend" list="#{'true':'是','false':'否'}" value=
"commend"> </s:radio>
所属性别：<s:select name="sexrequest" list="@com.lyq.model.Sex@getValues()"
   value="sexrequest.getName()"></s:select>
上传图片：<s:file id="file" name="file"></s:file>
商品说明：<s:textarea name="description" cols="50" rows="6"> </s:textarea>
```

（2）修改商品信息

用户编辑商品信息后单击“提交”按钮，将修改后的信息保存到数据库中。该操作发送一个 product_save.html 的 URL 请求，它会调用 ProductAction 类中的 save() 方法，上传图片并在数据表中添加数据，其实现代码如下：

```
public String save() throws Exception{
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    if(file != null ){                                          //如果文件路径不为空
    //获取服务器的绝对路径
String path = ServletActionContext.getServletContext().getRealPath ("/upload");
    File dir = new File(path);
    if(!dir.exists()){                                          //如果文件夹不存在
    dir.mkdir();                                                //创建文件夹
    }
    String fileName = StringUitl.getStringTime() + ".jpg";      //自定义图片名称
    FileInputStream fis = null;                                 //输入流
    FileOutputStream fos = null;                                //输出流
    try {
    fis = new FileInputStream(file);        //根据上传文件创建 InputStream 实例
    fos = new FileOutputStream(new File(dir,fileName));//创建写入服务器地址的输出流对象
    byte[] bs = new byte[1024 * 4];                             //创建字节数组实例
    int len = -1;
    while((len = fis.read(bs)) != -1){                          //循环读取文件
              fos.write(bs, 0, len);      //向指定的文件夹中写数据
              }
    UploadFile uploadFile = new UploadFile();                   //实例化对象
    uploadFile.setPath(fileName);                               //设置文件名称
    product.setUploadFile(uploadFile);                          //设置上传路径
    } catch (Exception e) {
    e.printStackTrace();
    }finally{
    fos.flush();
    fos.close();
    fis.close();
    }
    }
    //如果商品类别和商品类别 id 不为空，则保存商品类别信息
    if(product.getCategory() != null && product.getCategory().getId() != null){
    product.setCategory(categoryDao.load(product.getCategory().getId()));
    }
    //如果上传文件和上传文件 id 不为空，则保存文件的上传路径信息
    if(product.getUploadFile() != null && product.getUploadFile().getId() != null){
    product.setUploadFile(uploadFileDao.load(product.getUploadFile().getId()));
    }
productDao.saveOrUpdate(product);                               //保存商品信息
    return list();                                              //返回商品的查询方法
}
```

在 Web 应用中，文件上传通过 Form 表单实现，此时表单必须以 POST 方式提交（Struts 2 标签的 form 表单默认提交方式为 POST），并且设置 enctype ="multipart/form-data" 属性，在表单中需要提供一个或多个文件选择框供用户选择文件。提交表单后，选择的文件通过流方式传递，在接收表单的 Servlet 或 JSP 页面中获取该流并将流中的数据读到一个字节数组中。因此需要从中分离出每个文件的内容，并写到磁盘中。注意在分离过程中以字节为单位。

（3）删除商品信息

单击列表中的✕按钮，发送一个 product_del.html 的 URL 请求，触发 ProductAction 类中的 del() 方法。该方法将以商品的 id 为参数，执行持久化类中封装的 delete() 方法，调用 Hibernate 的 Session 对象中的 delete() 方法，其关键代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
public String del() throws Exception{
    productDao.delete(product.getId());                    //执行删除操作
    return list();                                         //返回商品列表查找方法
}
```

（4）添加商品信息

单击后台管理页面左侧导航栏中的“商品添加”超链接，打开“添加商品”页面，如图 19.36 所示。



图 19.36　“添加商品”页面

编辑商品信息后，单击“提交”按钮，发送一个 product_save.html 的 URL 请求，触发 ProductAction 类中的 save() 方法。

在 Struts 2 的后台 Action 配置文件 struts-admin.xml 中配置商品管理模块的 Action 及视图映射关系，关键代码如下：

```
<!-- 商品管理 -->
<package name="shop.admin.product" namespace="/admin/product" extends="shop.
admin">
    <action name="product_*" method="{1}" class="productAction">
    <result name="list">/WEB-INF/pages/admin/product/product_list.jsp</result>
    <result name="input">/WEB-INF/pages/admin/product/product_add.jsp</result>
    <result name="edit">/WEB-INF/pages/admin/product/product_edit.jsp</result>
    <interceptor-ref name="adminDefaultStack"/>
    </action>
</package>
```

19.11.2　实现商品类别管理功能

商品类别管理主要包括商品类别的查询、添加、修改及删除。

（1）查询商品类别

在后台的商品类别查询中通过树形下拉列表框形式显示给用户，如图 19.37 所示。

在生成商品类别树过程中，首先遍历每个根节点，然后采用迭代方式遍历每个根节点下的子节点，并将其放入以根节点 id 为 key 值的 Map 集合中，其流程如图 19.38 所示。



图 19.37　树形下拉列表框

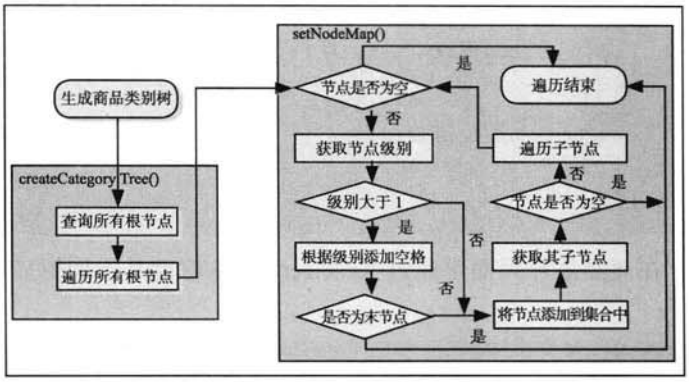


图 19.38　商品类别树生成流程

在打开商品页面的 edit() 方法中调用 createCategoryTree() 方法创建商品类别树，关键代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
private void createCategoryTree(){
 String where = "where level=1";                                    //查询一级节点
   PageModel<ProductCategory> pageModel = categoryDao.find(-1, -1,where ,null);
                                                                    //执行查询方法
   List<ProductCategory> allCategorys = pageModel.getList();
   map = new LinkedHashMap<Integer, String>();                      //创建新的集合
   for(ProductCategory category : allCategorys){                    //遍历所有的一级节点
     setNodeMap(map,category,false);                                //将其子节点添加到集合中
   }
}
```

在 setNodeMap() 方法中，首先判断节点是否为空。如果为空，则停止遍历。程序中根据获取的节点级别为类别名称添加字符串和空格，用于生成渐进的树形结构，将拼接后的节点放入 Map 集合中，并获取其子节点，重新调用 setNodeMap() 方法，直到遍历的节点为空为止，其关键代码如下：

```
private void setNodeMap(Map<Integer, String> map,ProductCategory node,boolean flag){
   if (node == null) {                                              //如果节点为空
   return;                                                          //返回空，结束程序运行
   }
   int level = node.getLevel();                                     //获取节点级别
   StringBuffer sb = new StringBuffer();                            //定义字符串对象
   if (level > 1) {                                                 //如果不是根节点
   for (int i = 0; i < level; i++) {
   sb.append("  ");                                                 //添加空格
   }
   sb.append(flag ? "├" : "└");          //如果为末节点则添加"└"，否则添加"├"
   }
   map.put(node.getId(), sb.append(node.getName()).toString());
                                                                    //将节点添加到集合中
   Set<ProductCategory> children = node.getChildren();
                                                                    //获取其子节点
   // 包含子类别
   if(children != null && children.size() > 0){ //如果节点不为空
   int i = 0;
   // 遍历子类别
   for (ProductCategory child : children) {
   boolean b = true;
   if(i == children.size()-1){                                      //如果子节点长度减 1 为 i，说明为末节点
             b = false;                                             //设置布尔常量为 false
             }
   setNodeMap(map,child,b);                                         //重新调用该方法
   }
   }
}
```

在商品添加页面中通过 <s:select> 标签将商品类别树显示在下拉列表框中，其关键代码如下：

```
<tr>
   <td width="105" height="22" bgcolor="#c6e8ff" align="right">选择类别：</td>
   <td><s:select list="map" name="category.id"></s:select></td>
</tr>
```

单击后台管理页面左侧导航栏中的“查询所有类别”超链接，发送一个 category_list.html

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

的 URL 请求，它触发 ProductCategoryAction 类中的 list() 方法，其关键代码如下：

```
public String list() throws Exception{
    Object[] params = null;                                    //对象数组为空
    String where;                                              //查询条件变量
    if(pid != null && pid > 0 ){                               //如果有父节点
    where = "where parent.id =?";                              //执行查询条件
    params = new Integer[]{pid};                               //设置参数值
    }else{
    where = "where parent is null";                            //查询根节点
    }
    pageModel = categoryDao.find(pageNo,pageSize,where,params);
                                                               //执行封装的查询方法
    return LIST;                                               //返回后台类别列表页面
}
```

该方法返回后台的商品类别列表页面，如图 19.39 所示。

（2）添加商品类别

单击导航栏中“添加商品类别”或商品类别列表页面中的“添加”超链接，打开“添加商品类别”页面，如图 19.40 所示。

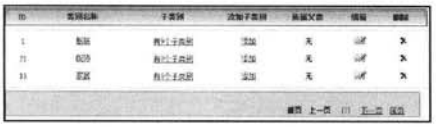


图 19.39　后台的商品类别列表页面



图 19.40　“添加商品类别”页面

输入类别名称后，单击“提交”按钮会触发 ProductCategoryAction 类中的 save() 方法，在其中判断该节点的父节点参数是否存在。如果存在，则设置其父节点属性，然后保存商品类别信息，其关键代码如下：

```
public String save() throws Exception{
    if(pid != null && pid > 0 ){                               //如果有父节点
    category.setParent(categoryDao.load(pid));                 //设置其父节点
    }
    categoryDao.saveOrUpdate(category);                        //添加类别信息
    return list();                                             //返回类别列表的查找方法
}
```

说明　商品类别列表页面中的“添加”超链接增加了一个参数 pid，单击这个超链接，该参数将传送给“添加商品类别”页面。在执行添加操作时该参数也会传送给 save() 方法。

（3）修改商品类别

单击商品类别列表中按钮，打开“编辑商品类别”页面，如图 19.41 所示。修改商品类别，单击“提交”按钮触发的也是 ProductCategoryAction 类的 save() 方法。



图 19.41　“编辑商品类别”页面

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（4）删除商品类别

单击“商品类别列表”页面中的✖按钮，发送一个 category_del.html 的 URL 请求，触发 ProductCategoryAction 类中的 del() 方法。该方法将以商品类别的 id 为参数，执行持久化类中封装的 delete() 方法，并删除指定的信息，其关键代码如下：

```
public String del() throws Exception{
    if(category.getId() != null && category.getId() > 0){
                                                    //判断是否获得 ID 参数
    categoryDao.delete(category.getId());           //执行删除操作
    }
    return list();                                  //返回商品类别列表的查找方法
}
```

在 Struts 2 的后台 Action 配置文件 struts-admin.xml 中配置商品类别管理模块的 Action 及视图映射关系，关键代码如下：

```
<!-- 类别管理 -->
<package name="shop.admin.category" namespace="/admin/product" extends= "shop.
admin">
    <action name="category_*" method="{1}" class="productCategoryAction">
    <result name="list">/WEB-INF/pages/admin/product/category_list.jsp</result>
    <result name="input">/WEB-INF/pages/admin/product/category_add.jsp</result>
    <result name="edit">/WEB-INF/pages/admin/product/category_edit.jsp</result>
    <interceptor-ref name="adminDefaultStack"/>
    </action>
</package>
```

19.12 后台订单管理模块设计

在后台的订单管理模块中，主要分为两个基本模块，分别是订单的查询和订单状态的修改，其中订单的查询又可分为订单的全部查询和用户自定义条件查询，框架模块图如图 19.42 所示。

19.12.1 实现后台订单查询

在管理页面左侧导航栏中单击“查看订单”超链接，打开订单状态管理页面，如图 19.43 所示。

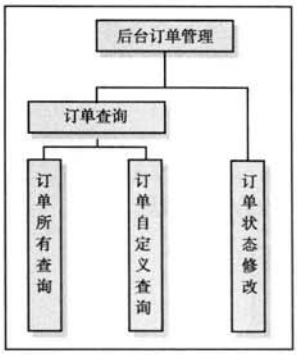


图 19.42 后台订单管理模块框架



图 19.43 订单状态管理页面

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

单击左侧导航栏中的“订单查询”超链接，打开“订单查询”页面，如图 19.44 所示。



图 19.44　“订单查询”页面

单击导航栏中的“查看订单”超链接或单击“订单查询”页面中的“提交”按钮均发送一个 order_list.html 的 URL 请求，触发 OrderAction 中的 list() 方法，其关键代码如下：

```
public String list() throws Exception {
    Map<String, String> orderby = new HashMap<String, String>(1);
                                                        //定义 Map 集合
    orderby.put("createTime", "desc");                  //设置按创建时间倒序排列
    StringBuffer whereBuffer = new StringBuffer("");    //创建字符串对象
    List<Object> params = new ArrayList<Object>();
    if(order.getOrderId() != null && order.getOrderId().length() > 0){
                                                        //如果订单号不为空
    whereBuffer.append("orderId = ?");                  //以订单号为查询条件
    params.add(order.getOrderId());                     //设置参数
    }
    if(order.getOrderState() != null){                  //如果订单状态不为空
    if(params.size() > 0) whereBuffer.append(" and ");  //增加查询条件
    whereBuffer.append("orderState = ?");               //设置订单状态为查询条件
    params.add(order.getOrderState());                  //设置参数
    }
    if(order.getCustomer() != null && order.getCustomer().getUsername() != null
    && order.getCustomer().getUsername().length() > 0){ //如果会员名不为空
    if(params.size() > 0) whereBuffer.append(" and ");  //增加查询条件
    whereBuffer.append("customer.username = ?");        //设置会员名为查询条件
    params.add(order.getCustomer().getUsername());      //设置参数
    }
    if(order.getName() != null && order.getName().length()>0){
                                                        //如果收款人姓名不为空
    if(params.size() > 0) whereBuffer.append(" and ");  //增加查询条件
    whereBuffer.append("name = ?");                     //设置收款人姓名为查询条件
    params.add(order.getName());                        //设置参数
    }
    //如果 whereBuffer 为空则查询条件为空，否则以 whereBuffer 为查询条件
    String where = whereBuffer.length()>0 ? "where "+whereBuffer.toString() : "";
    pageModel = orderDao.find(where, params.toArray(), orderby, pageNo, pageSize);
                                                        //执行查询方法
    return LIST;                                        //返回后台订单列表
}
```

“查看订单”超链接并没有为 list() 方法传递任何的参数，所以最后传给 find() 方法的 where 查询条件字符串为空，该方法将会从数据库中查询所有的订单信息并按创建时间的倒序输出。

list() 方法返回后台的订单信息列表页面，在其中利用 Struts 2 的 <s:iterator> 方法遍历输出返回结果集中的信息。后台订单信息列表页面的关键代码如下：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<table width="693" height="29" border="0" class="word01"><tr>
   <td width="140" align="center">订单号 </td>
   <td width="60" align="center">总金额 </td>
   <td width="63" align="center">消费者 </td>
   <td width="70" align="center">支付方式 </td>
   <td width="140" align="center">创建时间 </td>
   <td width="70" align="center">订单状态 </td>
   <td width="150" align="center">修改 </td>
   </tr></ table >
<table width="693" height="29" border="0">
   <s:iterator value="pageModel.list">
   <tr>
   <td width="140" align="center"><s:property value="orderId" /></td>
   <td width="60" align="center"><s:property value="totalPrice" /></td>
   <td width="63" align="center"><s:property value="customer.username" /></td>
   <td width="70" align="center"><s:property value="paymentWay.getName()" /></td>
   <td width="140" align="center">
          <s:date name="createTime" format="yyyy年MM月d日 HH:mm" /></td>
   <td width="70" align="center"><s:property value="orderState.getName()" /></td>
   <td width="150" align="center">
   <s:url action="order_select" namespace="/admin/product" ar="order_select">
   <s:param name="orderId" value="orderId"></s:param>
   </s:url> <input type="button" value="更新订单状态"
onclick="openWindow('${order_select}',350,150);"></td>
   </tr>
   </s:iterator>
</table>
```

19.12.2 实现后台订单状态管理

单击订单列表页面中的“更新订单状态”按钮，弹出提示对话框。让用户选择修改的状态信息，如图 19.45 所示。

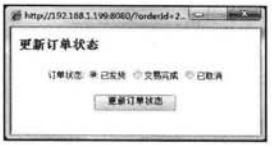


图 19.45 提示对话框

在订单列表页面中通过模态窗体形式弹出该对话框，为“更新订单状态”按钮绑定触发事件的关键代码如下：

```
<td width="150" align="center">
    <s:url action="order_select" namespace="/admin/product" var="order _select">
<s:param name="orderId" value="orderId"></s:param></s:url>
    <input type="button" value="更新订单状态"onclick="openWindow('$ {order_select}',
    350,150);">
</td>
```

说明 如果弹出的子窗体是模态的，则关闭后方可执行主窗体中的操作；如果是非模态的，不关闭也可执行主窗体中的操作。

Action 请求 order_select 跳转的页面为 order_select.jsp，即弹出的模态窗体，“更新订单状态”页面的关键代码如下：

```
<s:push value="order">
<h3>更新订单状态 </h3>
<div align="center">
<s:form action="order_update" namespace="/admin/product">
 <s:hidden name="orderId"></s:hidden>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<p>
订单状态：
    <s:radio name="orderState" list="@com.lyq.model.OrderState@getValues()"
value="orderState.getName()"></s:radio>
</p>
<s:submit value="更新订单状态" ></s:submit>
</s:form>
</div>
</s:push>
```

选择订单状态，单击“更新订单状态”按钮，发送一个 order_update.html 的 URL 请求，触发 OrderAction 类中的 update() 方法，其关键代码如下：

```
public String update() throws Exception {
   OrderState orderState = order.getOrderState();          //获取设置的订单状态
   order = orderDao.load(order.getOrderId());              //加载订单对象
   order.setOrderState(orderState);                        //设置的订单状态
   orderDao.update(order);                                 //修改订单状态
   return "update";                                        //返回订单状态修改成功页面
}
```

修改订单状态成功后，弹出提示窗口，如图 19.46 所示。

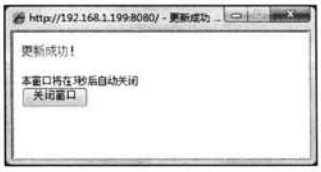


图 19.46　提示窗口

通过 JavaScript 设置该窗体 3 秒后自动关闭并刷新主页面。

设置窗体自动关闭的 JavaScript 关键代码如下：

```
<script type="text/javascript">
   function closewindow(){
   if(window.opener){
   window.opener.location.reload(true);                    //刷新父窗体
   window.close();                                         //关闭提示窗体
   }
}
function clock(){
   i = i-1;
   if(i > 0){                                              //如果 i 大于 0
   setTimeout("clock();",1000);                            //1 秒后重新调用 clock() 方法
   }else{
   closewindow();                                          //调用关闭窗体方法
   }
}
   var i = 3;                                              //设置 i 值
   clock();                                                //页面加载后自动调用 clock()
</script>
```

在上述代码中，通过变量 i 来设置窗体自动的关闭时间，在 clock() 方法中，当 i 值大于 0 时调用关闭窗体的方法，并且通过 setTimeout() 方法设置方法调用超时时间，参数 1000 的

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

单位为毫秒。

在 Struts 2 的后台 Action 配置文件 struts-admin.xml 中配置订单管理模块的 Action 及视图映射关系，关键代码如下：

```
<!-- 订单管理 -->
<package name="shop.admin.order" namespace="/admin/product" extends="shop.admin">
   <action name="order_*" method="{1}" class="orderAction">
   <result name="list">/WEB-INF/pages/admin/order/order_list.jsp</result>
   <result name="select">/WEB-INF/pages/admin/order/order_select.jsp</result>
   <result name="query">/WEB-INF/pages/admin/order/order_query.jsp</result>
   <result name="update">/WEB-INF/pages/admin/order/order_update_success.jsp</
   result>
   <interceptor-ref name="adminDefaultStack"/>
   </action>
</package>
```

19.13 网站发布

网站发布的目的是让浏览者在浏览器的地址栏中输入指定域名以访问网络。

19.13.1 配置 Tomcat 服务器

在发布之前需要配置 Tomcat 的端口号及虚拟主机，前者是为了用户在访问网站时不必输入端口号，后者是为了让 DNS 服务器解析 Tomcat 的域名信息。

（1）配置端口号

在安装 Tomcat 时默认端口号是 8080，需要修改为网络默认的端口号 80，即只要输入 Tomcat 服务器的 IP 地址即可访问网站。

例如，本机将 Tomcat 安装在 K:\Program Files 文件夹下。找到 K:\Program Files\Tomcat 7.0\conf\server.xml 文件，如图 19.47 所示。

通过记事本打开该文件，找到如下代码：

```
<Connector port="8080" protocol="HTTP/1.1"
maxThreads="150" connectionTimeout="20000"
 redirectPort="8443" />
```

修改 <Connector> 标签中 port 属性，将 8080 修改为 80，代码如下：

```
<Connector port="80" protocol="HTTP/1.1"
maxThreads="150" connectionTimeout="20000"
 redirectPort="8443" />
```

（2）配置虚拟主机

配置虚拟主机需要一个固定的 IP 地址，因为需要通过 DNS 服务器（域名服务器）映射这个 IP 地址到指定的域名空间。如果 IP 地址改变，DNS 服务器将无法映射指定的域名，这在很大程度上会影响用户正常访问网站或应用。

说明 虚拟主机是一种在一台 Web 服务器上服务多个域名的机制，每个域名都似乎独享了整台主机，大多数小型商务网站或应用的 Web 发布架设均采用了虚拟主机机制。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在配置虚拟主机前首先了解网上域名解析的简单流程，如图 19.48 所示。



图 19.47　需要修改的 xml 文件

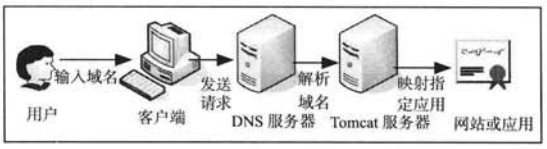


图 19.48　网上域名解析的简单流程

为了让 Tomcat 服务器映射到指定的应用，需要修改 server.xml 文件中的 Host 元素，在 XML 文件中找到如下代码：

```
<Host name="localhost"  appBase="webapps"
  unpackWARs="true" autoDeploy="true"
  xmlValidation="false" xmlNamespaceAware="false">
```

如 GO 购网络商城的域名为“www.mrshop.com”，webapp 文件夹中对应的应用文件夹为 Shop。设置虚拟主机映射的应用，修改如下：

```
<Host name="www.mrshop.com"  appBase="webapps/Shop"
  unpackWARs="true" autoDeploy="true"
  xmlValidation="false" xmlNamespaceAware="false">
```

说明　Tomcat 的 server.xml 文件在初始状态下只包括一台虚拟主机。设置多台虚拟主机只需要在该文件中添加多个 Host 元素。

这样在局域网中的机器即可通过“www.mrshop.com”地址访问 GO 购网络商城，为了让外网用户也能通过域名访问 GO 购网络商城，需要配置 DNS 服务器。

19.13.2　配置 DNS 服务器

本节将以 Windows Server 2003 为例介绍 DNS 服务器的配置。

在应用 DNS 服务器发布带域名的网站时，首先要确定服务器上已经安装了 DNS 服务，并且需要在 DNS 服务上新建一个区域和新建一个主机，然后配置 DNS 客户端。

（1）安装 DNS 服务

Step 1）单击“开始”/“控制面板”/“添加或删除程序”选项，打开“添加或删除程序”窗口，如图 19.49 所示。

Step 2）选择“添加 / 删除 Windows 组件”选项，打开“Windows 组件向导”对话框。选择“网络服务”复选框，如图 19.50 所示。

Step 3）单击“详细信息”按钮，打开如图 19.51 所示的“网络服务”对话框。选择安装“域名系统（DNS）”选项，单击“确定”按钮。



图 19.49　“添加或删除程序”窗口



图 19.50　选择“网络服务”复选框



图 19.51　“网络服务”对话框

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Step 4 单击“下一步”按钮，开始安装网络服务组件，如图 19.52 所示，在安装过程中需要将系统的安装盘插入到光驱中。

Step 5 单击“下一步”按钮，在弹出的“完成安装”对话框中单击“完成”按钮。

注意 安装 DNS 服务器的计算机必须有一个静态 IP 地址，此 IP 地址可以通过配置 TCP/IP 协议来获得。

（2）创建 DNS 区域

通过单击“开始”/“管理工具”/DNS 选项，打开 DNS 服务控制台，如图 19.53 所示。

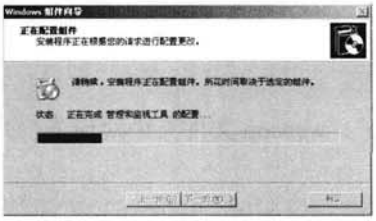


图 19.52 安装网络服务组件



图 19.53 DNS 服务控制台

Step 1 在左侧窗格中展开 MRKJ/“正向查找区域”选项，右击“正向查找区域”选项，在弹出的快捷菜单中选择“新建区域”选项，如图 19.54 所示。

Step 2 弹出“新建区域向导”对话框，单击“下一步”按钮，弹出“区域类型”对话框，如图 19.55 所示。

Step 3 选择区域类型后，单击“下一步”按钮，弹出“区域文件”对话框。在“创建新文件，文件名为”下的文本框中输入需要创建的新区域文件的名称，如图 19.56 所示。单击“下一步”按钮，弹出“动态更新”对话框，如图 19.57 所示。

Step 4 选择动态更新类型，单击“下一步”按钮，在弹出的对话框中单击“完成”按钮。

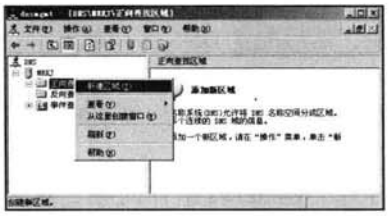


图 19.54 选择“新建区域”选项

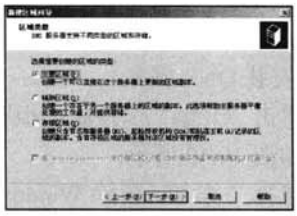


图 19.55 “区域类型”对话框

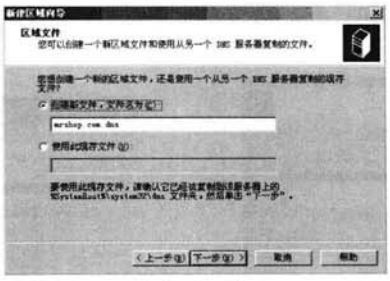


图 19.56 输入新区域文件的名称



图 19.57 “动态更新”对话框

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

说明　DNS 区域分为两类，一是正向搜索区域，即名称到 IP 地址的数据库，用于提供将名称转换为 IP 地址的服务；二是反向搜索区域，即 IP 地址到名称的数据库，用于提供将 IP 地址转换为名称的服务。

（3）新建主机

在多数情况下，DNS 客户端查询的是主机信息。例如，在正向搜索区域 mrshop.com（在配置 DNS 服务创建）中建立一个名为“www.mrshop.com”（在新建主机时创建）的主机资源记录，并且计算机的 IP 地址为 192.168.1.233（笔者的 IP 地址），配置方法如下。

Step 1）在 DNS 服务管理器中的左侧窗格中，右击“正向查找区域”下的 mrshop.com 区域，在弹出的快捷菜单中选择“新建主机”命令，如图 19.58 所示。



图 19.58　“新建主机”命令

Step 2）弹出“新建主机”对话框，在“名称”文本框中输入新建的主机名称，如“www”。注意，这是在通过 IE 地址栏浏览时需要输入的内容。

Step 3）在“IP 地址”文本框中输入主机对应的实际 IP 地址，如“192.168.1.233”，然后单击“添加主机”按钮。

说明　也可以在网上申请使用现有的 DNS 服务器。

（4）配置 DNS 客户端

DNS 客户端即 Tomcat 服务器所在的机器，以 Windows 7 系统为例，配置方法如下。

Step 1）右击客户端的“网络连接”图标，在弹出的快捷菜单中选择“属性”选项，弹出“本地连接 属性”对话框，如图 19.59 所示。

Step 2）选择“Internet 协议版本 4（TCP/IPv4）”选项，单击“属性”按钮，弹出如图 19.60 所示的“Internet 协议版本 4（TCP/IPv4）属性”对话框。

Step 3）在“使用下面的 DNS 服务器地址”中输入 DNS 服务器的 IP 地址，可以分别输入首选和备用 DNS 服务器的 IP 地址。一般 IP 都是指定的，主要用来解析域名的服务器。

Step 4）单击“确定”按钮。

Step 5）在客户端的 IE 地址栏中输入要访问的网址（如 www.mrshop.com）后即可打开 GO 购网络商城网站首页，如图 19.61 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 19.59 “本地连接 属性”对话框



图 19.60 “Internet 协议版本 4（TCP/IPv4）属性”对话框



图 19.61 GO 购网络商城首页

19.14 开发常见问题与解决

19.14.1 页面中出现中文乱码

问题描述：在程序测试的过程中发现，页面中会出现程序乱码的情况。

解决方法：要解决 Struts 2 的乱码问题可以在 struts.properties 文件中进行如下配置。

```
struts.i18n.encoding=UTF-8
```

struts.i18n.encoding 用来设置 Web 的默认编码方式，GO 购网络商城使用了 UTF-8 作为默认的编码方式，虽然该方法可以有效解决表单的中文乱码问题，但是该模式要求表单的 method 属性必须为 post，由于 Struts 2 中的 form 表单标签默认的 method 属性就为 post，所以不必再进行额外的设置，如果页面中的表单没有使用 Struts 2 的表单标签，则需要在表单中指定 method 的属性值。

19.14.2 自定义查询时，当订单号为空会出错

问题描述：在对后台订单自定义查询进行测试的时候发现，如果订单号为空，根据填写后的查询条件进行查询就会报错，如图 19.62 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

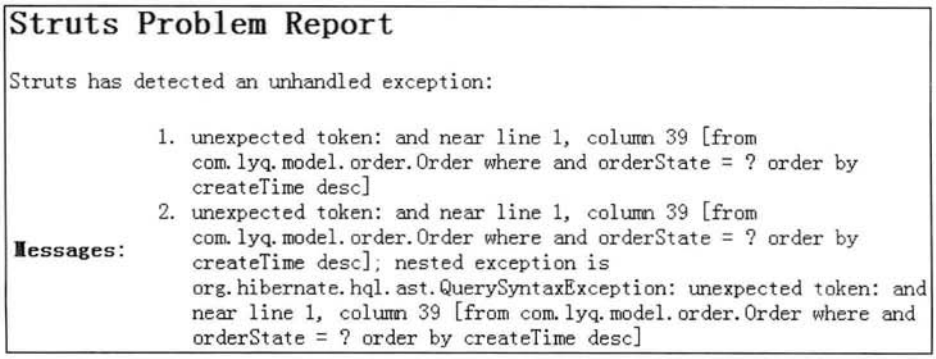


图 19.62　订单号为空查询时系统的报错信息

解决方法：从错误信息中可以发现，where 条件查询语句中的关键字 where 直接连接的是关键字 and，也就是说在订单状态前置查询条件订单号码为空的情况下，同样将关键字 and 拼接到了查询条件中，解决该问题只需对查询条件的 Object 数组的长度进行判断，如果数组的长度不为 0，就将关键字 and 添加到查询条件中，以订单状态的查询条件为例，其关键代码如下：

```
if(order.getOrderState() != null){                    // 如果订单状态不为空
    if(params.size() > 0){
    whereBuffer.append(" and ");                      // 增加查询条件
    }
    ...                                               // 省略的代码
}
```

19.15　小结

本章重点讲解了 GO 购网络商城中关键模块的开发过程以及网站的发布方法。通过对本章的学习，读者应该能够熟悉软件的开发流程，并重点掌握如何配置 Struts 2+Hibernate 4+Spring 3 的开发环境。另外，读者还应该掌握如何实现数据分页，以及购物车的实现流程。购物车功能是网络商城中必不可少的功能模块，读者要重点掌握。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Java学习群

72030155

进群可以免费获取视频教程以及每日免费听老师讲课

Java学习群：72030155

好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。